ORE Product Catalogue

Payoffs, Trade Input, Pricing

24 June 2025

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
September 2024	Acadia	updates for release 13
January 2025	Acadia	separate user guide, risk methodology, product catalogue

Contents

1 Introduction

The ORE documentation comes in three parts

- User Guide: ORE overview and history, discussion of the usage examples that come with ORE, "static" parameterisation reference, market data reference
- Methodology: Overview over the risk methods applied in ORE
- Products: Product catalogue with a full description of instrument payoffs, pricing methodology and ORE XML input guide

This document is the third part. It starts with descriptions of payoffs and ORE XML input rules for all products covered by ORE, followed by a similar length elaboration of the pricing methodology applied per product.

2 Trade Data

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

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

An example of a `portfolio.xml` file with one Swap trade including the full envelope node is shown in Listing 1.

Listing 1: Portfolio

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

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

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

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

2.1 Envelope

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

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

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

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

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

- `NettingSetId` [Optional]: The `NettingSetId` element specifies the identifier for a netting set. If a `NettingSetId` is specified, the trade is eligible for close-out netting under the terms of an associated ISDA agreement. The specified `NettingSetId` must be defined within the netting set definitions file (see section 3). If left blank or omitted the trade will not belong to any netting set, and thus not be eligible for netting.

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

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

  Allowable values for each PortfolioId: Any string.

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

  Allowable values: Any custom element.

2.1.1 Netting Set Details

Instead of a single netting set ID, defined by a `NettingSetId` node, an alternative `NettingSetDetails` node can be provided, which itself contains a `NettingSetId` sub-node, and four other optional sub-nodes, which altogether allow for extending the uniqueness of netting sets beyond the netting set ID. The allowable values for each sub-node are any alphanumeric string. The underscore ('_') sign may be used as well.

The `NettingSetDetails` node is given in the following XML format:

Listing 2: Netting set details

```
<NettingSetDetails>
    <NettingSetId> </NettingSetId>
    <AgreementType> </AgreementType>
    <CallType> </CallType>
    <InitialMarginType> </InitialMarginType>
    <LegalEntityId> </LegalEntityId>
</NettingSetDetails>
```

2.2 Trade Specific Data

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

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

2.2.1 Swap

Payoff

An interest rate swap (**IRS**) is an agreement between two counterparties in which one stream of future interest payments (leg) is exchanged for another based on a specified notional amount. A vanilla interest rate swap involves two legs in the same currency, exchanging a floating rate benchmarked to an Interbank Offered Rate (IBOR)[1] index of a specified tenor for a fixed rate, or vice versa.

- The notional amount for each leg may be fixed, amortising or accreting.
- The fixed rate and the spread can vary over the lifetime of the swap.

A single currency basis swap (**BS**) has two floating legs, benchmarked to any two of the supported IBOR indices. The two legs may also have different tenors of the same IBOR index.

- The notional amount for each leg may be fixed, amortising or accreting.
- The floating leg spreads can vary over the lifetime of the swap.

An Overnight Index Swap (**OIS**) has at least one leg where the floating rate is benchmarked to an overnight index rate, typically the rate for overnight unsecured lending between banks.

- For example, in USD the overnight index rate would be the Federal Funds rate, in EUR it would be Eonia, and in GBP it would be Sonia.
- The overnight index rate leg compounds on a daily basis as per the corresponding overnight rate.

The typical OIS Swap exchanges compounded overnight interest for a fixed leg, paying once at maturity for maturities up to a year and paying annually for longer maturities. In the US OIS Swap market another flavour is common that exchanges an OIS linked leg paying the arithmetic weighted average of O/N fixings (quarterly, without compounding) for a floating USD-LIBOR-3M linked leg with a spread.

A Bond Market Association (**BMA**) Swap has at least one leg where the floating rate is benchmarked to the BMA's floating rate municipal swap index. That is, the floating leg rate is based upon fixings of the US SIFMA Municipal Swap Index (formerly the

[1] IBOR is a generic term for a family of indices that represent a rate for deposit periods, or tenors, longer than one day. Specific IBOR indices are usually named after the location where they are set, such as LIBOR (London), TIBOR (Tokyo) or EURIBOR (Eurozone).

BMA Municipal Index or "BMA Index"). The non-BMA leg of a BMA Swap can be fixed or floating.

A Constant Maturity Swap (**CMS**) has at least one leg where the floating rate is benchmarked against a CMS index using the market rate of a fixed maturity instrument, such as a swap, with a longer maturity than the length of the reset period. The CMS leg may contain a cap, floor, or collar.

Similar to a CMS, a variable rate Swap leg can be linked to a Constant Maturity Bond (**CMB**) index. In contrast to CMS the yield is determined by the yields of a class of Bonds, such as Government Bonds. As in the CMS case, the term of the yield is typically longer than the length of the reset period. A CMB index in ORE is of the form CMB-FAMILY-TENOR. Note, that the CMB linked legs in ORE currently do not support Caps, Floors or Collars.

Input

The `SwapData` node is the trade data container for the *Swap* trade type. A Swap must have at least one leg, and can have an unlimited number of legs. Each leg is represented by a `LegData` trade component sub-node, described in section 2.3.3. An example structure of a two-legged `SwapData` node is shown in Listing 3.

- Settlement [Optional]: Delivery type applicable to cross currency swaps, and ignored for all other swap types. Delivery type does not impact pricing in ORE, but npv results are produced with and without SIMM exemptions.

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

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

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

Listing 3: Swap data

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

Note that Swaps in non-deliverable currencies with payment in a deliverable currency are supported by setting Settlement to *Cash* and - on both legs - using the Indexings node (2.3.8), as well as setting the Currency to the deliverable currency, while keeping the Notional expressed in the non-deliverable currency amount.

Within the Indexings node, an fx Index field is mandatory defining the deliverable and non-deliverable currencies and fixing source. The Indexing node can also include optional FixingCalendar, IsInArrears and FixingDays fields to determine the date(s) of the fx fixing(s). See Listing 4 for an example non-deliverable IR swap where USD is the payment currency and CLP is the non-deliverable currency.

Listing 4: Non deliverable single currency IR Swap

```
<SwapData>
  <Settlement>Cash</Settlement>
  <LegData>
   <LegType>Fixed</LegType>
   <Payer>false</Payer>
   <Currency>USD</Currency><!-- Payment currency is USD rather than CLP -->
    <Notionals>
     <Notional>850000000</Notional><!-- in CLP -->
    </Notionals>
    <Indexings>
      <Indexing>
        <Index>FX-TR20H-CLP-USD</Index><!-- to convert CLP flows into USD -->
        <FixingCalendar>CLP,USD</FixingCalendar>
        <IsInArrears>true</IsInArrears>
        <FixingDays>2</FixingDays>
      </Indexing>
    </Indexings>
    ...
  </LegData>
  <LegData>
   <LegType>Floating</LegType>
   <Payer>true</Payer>
   <Currency>USD</Currency><!-- Payment currency is USD rather than CLP -->
    <Notionals>
     <Notional>850000000</Notional><!-- in CLP -->
     </Notionals>
     <Indexings>
      <Indexing>
        <Index>FX-TR20H-CLP-USD</Index><!-- to convert CLP flows into USD -->
        <FixingCalendar>CLP,USD</FixingCalendar>
        <IsInArrears>true</IsInArrears>
        <FixingDays>2</FixingDays>
      </Indexing>
     </Indexings>
    ...
  </LegData>
</SwapData>
```

2.2.2 Zero Coupon Swap

Payoff

A Zero-Coupon Swap has at least one zero-coupon leg. This leg has no interest rate coupon payments during the life of the swap, just one final interest payment at maturity, akin to a zero-coupon bond. The zero-coupon leg compounds at the tenor of the swap.

Input

A Zero Coupon swap is set up as a swap (trade type *Swap*) , with one leg of type `ZeroCouponFixed`. Listing 5 shows an example. The ZeroCouponFixed leg contains an additional `ZeroCouponFixedLegData` block. See 2.3.19 for details on the ZeroCouponFixed leg specification.

Listing 5: Zero Coupon Swap Data

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

2.2.3 Cap/Floor

Payoff

A single floating swap leg with interest payments benchmarked to IBOR or CMS indices, may have a cap, floor, or collar. This creates a series of European interest rate options (caplets or floorlets) where the cap or floor rate is the strike price, and each floating rate reset date is an option expiry date.
A collar is the combination of a series of long caplets and short floorlets for a long position in the collar.
Payoff of a caplet on an IBOR floating leg coupon period to the payer of the leg (the option buyer):

$$N[L(t_{i-1}, t_i) - K]^+ \delta(t_{i-1}, t_i) = N \max(0, L(t_{i-1}, t_i) - K)\, \delta(t_{i-1}, t_i)$$

Payoff of a floorlet on an IBOR floating leg coupon period to the receiver of the leg (the option buyer):

$$N[K - L(t_{i-1}, t_i)]^+ \delta(t_{i-1}, t_i) = N \max(0, K - L(t_{i-1}, t_i))\, \delta(t_{i-1}, t_i)$$

Where:

- N: the Notional
- $L(t_{i-1}, t_i)$: the IBOR index fixing for the period
- K: the cap or floor rate, i.e. the strike

- $\delta(t_{i-1}, t_i)$: the daycount fraction for the coupon period

For a caplet or floorlet on a CMS index the $L(t_{i-1}, t_i)$ index can be replaced by CMS_n where n indicates CMS rate for n number of years.

Standalone caps and floors with a single payoff on an IBOR or CMS index are also supported. Payoff of a cap/floor on an IBOR index:

$$N \cdot [\omega(L(t_{i-1}, t_i) - K)]^+$$

where $\omega = 1$ for a cap, $\omega = -1$ for a floor.
Payoff of a cap/floor on a CMS index:

$$N \cdot [\omega(CMS_n - K)]^+$$

where CMS_n is the n year CMS rate.

Input

The `CapFloorData` node is the trade data container for the *CapFloor* trade type. It's a cap, floor or collar (i.e. a portfolio of a long cap and a short floor for a long position in the collar) on a series of Ibor, SIFMA, OIS, CMS, Duration-adjusted CMS, CMS Spread, CPI, YY coupons.

The `CapFloorData` node contains a `LongShort` sub-node which indicates whether the cap (floor, collar) is long or short, and a `LegData` sub-node where the LegType can be set to *Floating*, *CMS*, *CMSSpread*, *DurationAdjustedCMS*, *CPI* or *YY*, plus elements for the Cap and Floor rates. An example structure with Cap rates is shown in in Listing 6. The optional node *PaymentDates* in the `LegData` subnode is currently only used for OIS and IBOR indices (see 2.3.3).

A `CapFloorData` node must have either `Caps` or `Floors` elements, or both. In the case of both (I.e. a collar with long cap and short floor) the sequence is that `Caps` elements must be above the `Floors` elements. Note that the `Caps` and `Floors` elements must be outside the `LegData` sub-node, i.e. a *CapFloor* can't have a capped or floored *Floating* or *CMS* leg. The *Payer* flag in the LegData subnode is ignored for this instrument. Notice that the signs in the definition of a collar (long cap, short floor) for the CapFloor instruments is exactly opposite to 2.3.6.

Listing 6: Cap/Floor data

```
<CapFloorData>
  <LongShort>Long</LongShort>
  <LegData>
    <Payer>false</Payer>
    <LegType>Floating</LegType>
     ...
  </LegData>
  <Caps>
    <Cap>0.05</Cap>
  </Caps>
  <Premiums>
    <Premium>
      <Amount>1000</Amount>
      <Currency>EUR</Currency>
      <PayDate>2021-01-27</PayDate>
    </Premium>
  </Premiums>
</CapFloorData>
```

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

- LongShort: This node defines the position in the cap (floor, collar) and can take values *Long* or *Short*.
- LegData: This is a trade component sub-node outlined in section 2.3.3. Exactly one `LegData` node is allowed, and the LegType element must be set to *Floating* (Ibor and OIS), *CMS*, *CMSSpread*, *DurationAdjustedCMS*, *CPI* or *YY*.
- Caps: This node has child elements of type `Cap` capping the floating leg (after applying spread if any). The first rate value corresponds to the first coupon, the second rate value corresponds to the second coupon, etc. If the number of coupons exceeds the number of rate values, the rate will be kept flat at the value of last entered rate for the remaining coupons. For a fixed cap rate over all coupons, one single rate value is sufficient. The number of entered rate values cannot exceed the number of coupons.

  Allowable values for each `Cap` element: Any real number. The rate is expressed in decimal form, eg 0.05 is a rate of 5%
- Floors: This node has child elements of type `Floor` flooring the floating leg (after applying spread if any). The first rate value corresponds to the first coupon, the second rate value corresponds to the second coupon, etc. If the number of coupons exceeds the number of rate values, the rate will be kept flat at the value of last entered rate for the remaining coupons. For a fixed floor rate over all coupons, one single rate value is sufficient. The number of entered rate values cannot exceed the number of coupons.

  Allowable values for each `Floor` element: Any real number. The rate is expressed in decimal form, eg 0.05 is a rate of 5%
- Premiums [Optional]: Option premium amounts paid by the option buyer to the

option seller.

Allowable values: See section 2.3.2

2.2.4 Forward Rate Agreement

Payoff

A Forward Rate Agreement is a contract between two counterparties that determines a fixed forward interest rate benchmarked to a reference IBOR index rate and tenor.

- Difference between the forward rate and the reference rate is to be paid/received at the end of the contract.
- Interest is accrued on a predetermined notional amount from an agreed upon start date to the end of the FRA contract.
- There are no exchanges of notional.

Input

A forward rate agreement (trade type *ForwardRateAgreement* is set up using a `ForwardRateAgreementData` block as shown in listing 7. The forward rate agreement specific elements are:

- StartDate: A FRA expires/settles on the startDate.
  Allowable values: See `Date` in Table 13.
- EndDate: EndDate is the date when the forward loan or deposit ends. It follows that (EndDate - StartDate) is the tenor/term of the underlying loan or deposit.

  Allowable values: See `Date` in Table 13.
- Currency: The currency of the FRA notional.
  Allowable values: See Table 15 `Currency`.
- Index: The name of the interest rate index the FRA is benchmarked against.

  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 19.
- LongShort: Specifies whether the FRA position is long (one receives the agreed rate) or short (one pays the agreed rate).

  Allowable values: *Long*, *Short*.
- Strike: The agreed forward interest rate.

  Allowable values: Any real number. The strike rate is expressed in decimal form, e.g. 0.05 is a rate of 5%.
- Notional: No accretion or amortisation, just a constant notional.
  Allowable values: Any positive real number.

Listing 7: Forward Rate Agreement Data

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

2.2.5 Swaption

Payoff

A swaption is an option which gives the buyer the right, but not the obligation, to enter into an underlying interest rate swap. The underlying swap may have varying notional, rates, and spreads during its lifetime.

With a **European Swaption**, the buyer is only allowed to exercise the option and enter into the swap on the expiration date of the swaption. There are two types of swaptions, payer and receiver:

- Payer swaptions: the underlying swap pays fixed rate and receives floating rate.
- Receiver swaptions: the underlying swap receives fixed and pays floating rate.

European Swaption settlement can be either cash or physical delivery of the underlying swap. For cash settlement there are different methods to compute the settlement price (collateralized cash price, par yield curve).

In a **Bermudan Swaption**, the buyer is allowed to exercise the option and enter into the underlying swap on a predetermined set of dates rather than a single expiration date. The underlying swap may have varying notional, rates, and spreads during its lifetime.

For an **American Swaption** the buyer is allowed to exercise the option any time between two specified dates.

Input

The `SwaptionData` node is the trade data container for the *Swaption* trade type. The `SwaptionData` node has one and exactly one `OptionData` trade component sub-node, and at least one `LegData` trade component sub-node. These trade components are outlined in section 2.3.1 and section 2.3.3.

Supported swaption exercise styles are *European*, *Bermudan*, *American*. Swaptions of all exercise styles can have an arbitrary number of legs, with each leg represented by a `LegData` sub-node. Cross currency swaptions are not supported for either exercise style, i.e. the Currency element must have the same value for all `LegData` sub-nodes of a swaption. There must be at least one full coupon period after the exercise date for European Swaptions, and after the last exercise date for Bermudan and American Swaptions. See Table 1 for further details on requirements for swaptions.

The structure of an example `SwaptionData` node of a European swaption is shown in Listing 8.

Listing 8: Swaption data

```
<SwaptionData>
    <OptionData>
        <LongShort>Long</LongShort>
        <Style>European</Style>
        <Settlement>Physical</Settlement>
        <ExerciseDates>
          <ExerciseDate>2027-03-02</ExerciseDate>
        </ExerciseDates>
        ...
        <Premiums>
          <Premium>
            <Amount>807000</Amount>
            <Currency>GBP</Currency>
            <PayDate>2021-06-15</PayDate>
          </Premium>
        </Premiums>
    </OptionData>
    <LegData>
        <LegType>Fixed</LegType>
        <Payer>false</Payer>
        <Currency>GBP</Currency>
        ...
    </LegData>
    <LegData>
        <LegType>Floating</LegType>
        <Payer>true</Payer>
        <Currency>GBP</Currency>
        ...
    </LegData>
</SwaptionData>
```

	A Swaption requires:
`OptionData`	One `OptionData` sub-node
`Style`	*Bermudan* or *European* or *American*
`ExerciseDates`	*European* swaptions can only have one `ExerciseDate` child element. *American* swaptions must have two `ExerciseDate` child elements. *Bermudan* swaptions must have at least two `ExerciseDate` child elements, or a Rules or Dates based exercise schedule.
`LegData`	At least one `LegData` sub-node
`Currency`	The same currency for all `LegData` sub-nodes.
`LegType`	Allowed types are *Cashflow*, *Fixed* or *Floating*. Floating coupons can be (capped / floored) Ibor, (capped / floored) compounded or averaged OIS, or BMA/SIFMA. Standalone options (nakedOption = true) are not allowed, neither are local OIS cap/floors.

Table 1: Requirements for Swaptions

The `OptionData` trade component sub-node is outlined in section 2.3.1. The relevant fields in the `OptionData` node for a Swaption are:

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

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

Table 2: Swaption set up and resulting flows

Note that for CallableSwaps, contrary to the above, the payer and receiver legs in the underlying swap are instead from the perspective of the client, like for a standalone Swap trade.

- `OptionType`[Optional]: This flag is optional for swaptions, and even if set, has no impact. Whether a swaption is a payer or receiver swaption is determined by the Payer flags on the legs of the underlying swap.
- `Style`: The exercise style of the Swaption. The allowable values are *European*, *Bermudan* or *American*.
- `NoticePeriod`[Optional]: The notice period defining the date (relative to the exercise date) on which the exercise decision has to be taken. If not given the notice period defaults to *0D*, i.e. the notice date is identical to the exercise date. Allowable values: A number followed by *D, W, M, or Y*
- `NoticeCalendar`[Optional]: The calendar used to compute the notice date from the exercise date. If not given defaults to the *NullCalendar* (no holidays, weekends are no holidays either). Allowable values: See Table 17 `Calendar`.
- `NoticeConvention`[Optional]: The roll convention used to compute the notice date from the exercise date. Defaults to *Unadjusted* if not given. Allowable values: See Table 14 Roll Convention.

- `Settlement`: Delivery Type. The allowable values are *Cash* or *Physical*. Note that for TradeType *CallableSwap* only *Physical* is allowed. Also note that *Cash* means the last Exercise date will be the Maturity date, whereas for *Physical* the Maturity date will be the End date of the underlying Swap.

- `SettlementMethod`[Optional]: Specifies the method to calculate the settlement amount for Swaptions and CallableSwaps. Allowable values: *PhysicalOTC*, *PhysicalCleared*, *CollateralizedCashPrice*, *ParYieldCurve*. Defaults to *ParYieldCurve* if Settlement is *Cash* and defaults to *PhysicalOTC* if Settlement is *Physical*.

  PhysicalOTC = OTC traded swaptions with physical settlement
  PhysicalCleared = Cleared swaptions with physical settlement
  CollateralizedCashPrice = Cash settled swaptions with settlement price calculation using zero coupon curve discounting
  ParYieldCurve = Cash settled swaptions with settlement price calculation using par yield discounting [2] [3]

- `MidCouponExercise` [Optional]: If *false*, the exercise-into underlying comprises all coupons with accrual start date greater or equal to notification date. I.e. one exercises into the next coupon, not the current one.
  If *true*, the exercise-into underlying comprises all coupons with accrual end date greater than the effective exercise date which is computed from the notification date by adding the notice period. The accrual paid for such coupons on exercise is calculated from the effective exercise date to the accrual end date (short coupon).

  Allowable values: *true*, *false*. If omitted, defaults to *false* for *European* and *Bermudan* swaptions and *true* for *American* swaptions.

- `ExerciseFees`[Optional]: This node contains child elements of type `ExerciseFee`. Similar to a list of notionals (see 2.3.3) the fees can be given either

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

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

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

  Only supported for Swaptions and Callable Swaps currently.

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

- `ExerciseFeeSettlementPeriod`[Optional]: The settlement lag for exercise fee payments. Defaults to 0D if not given. This lag is relative to the exercise date (as opposed to the notice date). Allowable values: A number followed by *D, W, M, or Y*
- `ExerciseFeeSettlementCalendar`[Optional]: The calendar used to compute the exercise fee settlement date from the exercise date. If not given defaults to the *NullCalendar* (no holidays, weekends are no holidays either). Allowable values: See Table 17 Calendar.
- `ExerciseFeeSettlementConvention`[Optional]: The roll convention used to compute the exercise fee settlement date from the exercise date. Defaults to *Unadjusted* if not given. Allowable values: See Table 14 Roll Convention.
- An `ExerciseDates` node where for *European* style swaptions exactly one `ExerciseDate` date element must be given, and for *American* style swaptions exactly two `ExerciseDate` date element must be given, defining the start and the end of the American exercise period. *Bermudan* style swaptions can have `ExerciseDate` elements given directly (at least two `ExerciseDate` elements must be given). See Listing 9
- *Bermudan* style swaptions can also have Rules or Dates based exercise dates using an `ExerciseSchedule` node instead of `ExerciseDates`. See Listings 10 and 11.

Listing 9: Bermudan Swaption ExerciseDate:s

```
<SwaptionData>
    <OptionData>
        <LongShort>Long</LongShort>
        <Style>Bermudan</Style>
        <Settlement>Physical</Settlement>
        <ExerciseDates>
          <ExerciseDate>2027-03-02</ExerciseDate>
          <ExerciseDate>2028-03-02</ExerciseDate>
          <ExerciseDate>2029-03-02</ExerciseDate>
        </ExerciseDates>
        ...
    </OptionData>
   ...
```

Listing 10: Bermudan Swaption Rules based

```
<SwaptionData>
    <OptionData>
        <LongShort>Long</LongShort>
        <Style>Bermudan</Style>
        <Settlement>Physical</Settlement>
        <ExerciseSchedule>
         <Rules>
          <StartDate>2027-03-02</StartDate>
          <EndDate>2029-03-02</EndDate>
          <Tenor>1Y</Tenor>
          <Calendar>US</Calendar>
          <Convention>MF</Convention>
         </Rules>
        </ExerciseSchedule>
        ...
    </OptionData>
   ...
```

Listing 11: Bermudan Swaption Dates based

```
<SwaptionData>
    <OptionData>
        <LongShort>Long</LongShort>
        <Style>Bermudan</Style>
        <Settlement>Physical</Settlement>
        <ExerciseSchedule>
         <Dates>
          <Calendar>NullCalendar</Calendar>
          <Convention>Unadjusted</Convention>
          <Dates>
           <Date>2027-03-02</Date>
           <Date>2028-03-02</Date>
           <Date>2029-03-02</Date>
          </Dates>
         </Dates>
        </ExerciseSchedule>
        ...
    </OptionData>
   ...
```

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

  Allowable values: See section 2.3.2

- An `ExerciseData` [Optional] node where one `Date` element must be given, and one `Price` element can optionally also be given. See Listing 12

  This node marks the Swaption as exercised. If the `ExerciseData` node is omitted it is assumed the Swaption has not been exercised.

  The effective exercise date is the next `ExerciseDate` in the `ExerciseDates` node greater or equal the given `Date` in `ExerciseData`.

For a cash-settled Swaption, the `Price` given in `ExerciseData` represents the cash settlement amount. It is paid according to the `PaymentData` node: If an explicit list of payment dates is given, the payment takes place on the next date following the effective exercise date. If the `PaymentData` is rules-based, the payment date is derived from the effective exercise date using the given calendar, lag and convention.

If a Swaption is cash-settled and has an `ExerciseData` node with a `Date` but no `Price`, then the Swaption is considered exercised on the given date, but without a settlement amount being paid.

Listing 12: ExerciseData to mark a Swaption or CallableSwap as exercised

```
<ExerciseData>
  <Date>2023-09-03</Date>
  <Price>112000</Price>
</ExerciseData>
```

- A `PaymentData` [Optional] node can be added which defines dates or rules-based settlement date(s) for cash-settled Swaptions. Note that if rules-based, only *Exercise* is allowed in the `RelativeTo` field for Swaptions. See `PaymentData` in 2.3.1

2.2.6 Callable Swap

Payoff

A Callable Swap can be terminated by one of the parties on specific dates and thus can be decomposed into a swap and a physically settled swaption, the latter representing the call right. The underlying swap must be fixed versus float and may have varying notional, rates, and spreads during its lifetime.

Input

The `CallableSwapData` node is the trade data container for the *CallableSwap* trade type. A Callable Swap is a swap that can be cancelled at predefined dates by one of the counterparties. A Callable Swap must have at least one leg, each leg described by a `LegData` trade component sub-node as described in section 2.3.3.

Unless MidCouponExercise is *true*, there must be at least one full coupon period after the exercise date for European Callable Swaps, and after the last exercise date for Bermudan and American Callable Swaps.

The `CallableSwapData` node also contains an `OptionData` node which describes the exercise dates and specifies which party holds the call right, see 2.3.1. An example structure of a `CallableSwapData` node is shown in Listing 13.

Listing 13: Callable Swap data

```
<CallableSwapData>
    <OptionData>
      <LongShort>Short</LongShort>
      <Style>Bermudan</Style>
      <Settlement>Physical</Settlement>
      <MidCouponExercise>true<MidCouponExercise>
      <ExerciseDates>
        <ExerciseDate>2031-10-01</ExerciseDate>
        <ExerciseDate>2032-10-01</ExerciseDate>
        <ExerciseDate>2033-10-01</ExerciseDate>
      </ExerciseDates>
      ...
    </OptionData>
  <LegData>
        <LegType>Fixed</LegType>
        <Payer>false</Payer>
        <Currency>USD</Currency>
        ...
  </LegData>
  <LegData>
        <LegType>Floating</LegType>
        <Payer>true</Payer>
        <Currency>USD</Currency>
        ...
  </LegData>
</CallableSwapData>
```

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

- OptionData: This is a trade component sub-node outlined in section 2.3.1. The exercise dates specify the dates on which one of the counterparties may terminate the swap. The counterpart holding the call right is specified by the `LongShort` flag. The Settlement should be set to *Physical* always. See also the OptionData node outlined for a Swaption - see 2.2.5, which is identical for a CallableSwap with the exception of the requirement that Settlement must be *Physical*, and that the leg directions on a CallableSwap are from the perspective of the client, whereas they are from the perspective of the party that is long on a Swaption. A callable swap can be marked as exercised as explained in 2.2.5 using the `ExerciseData` node within OptionData.
- LegData: This is a trade component sub-node described in section 2.3.3 outlining each leg of the underlying Swap. A Callable Swap must have at least one leg on the underlying Swap, but can have multiple legs, i.e. multiple `LegData` nodes. The LegType elements must be of types *Floating*, *Fixed* or *Cashflow*. All legs must have the same `Currency`.

  Note that the direction of the legs, determined by the `Payer` tag, is like for a Swap, from the perspective of the party to the trade. I.e. unlike for a Swaption where the direction of the legs is from the perspective of the party that is long.

2.2.7 Cash Position

The `CashPositionData` node is the trade data container for the *CashPosition* trade type. The structure - including example values - of the `CashPositionData` node is shown in Listing 14.

A cash position can be used both as a stand alone trade type (TradeType: *CashPosition*) or as a trade component within the *TotalReturnSwap* (Generic TRS) trade type.

Listing 14: Cash Position data

```
<CashPositionData>
    <Currency>EUR</Currency>
    <Amount>1000000</Amount>
</CashPositionData>
```

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

- Currency: The currency of cash position.
  Allowable values: See Table 15 `Currency`.
- Amount: The amount of cash position.
  Allowable values: Any real number.

2.2.8 FX Forward

Payoff

An FX forward is a contract that locks in the FX rate for the exchange of a set amount of one currency for another at a predetermined time in the future. An FX Forward does not involve any upfront payment.

Input

The `FXForwardData` node is the trade data container for the *FxForward* trade type. The structure - including example values - of the `FXForwardData` node is shown in Listing 15.

Listing 15: FX Forward data

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

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

- ValueDate: The value date of the FX Forward.
  Allowable values: See `Date` in Table 13.
- BoughtCurrency: The currency to be bought on value date.
  Allowable values: See Table 15 `Currency`.
- BoughtAmount: The amount to be bought on value date.
  Allowable values: Any positive real number.
- SoldCurrency: The currency to be sold on value date.
  Allowable values: See Table 15 `Currency`.
- SoldAmount: The amount to be sold on value date.
  Allowable values: Any positive real number.
- Settlement [Optional]: Delivery type. Note that Non-Deliverable Forwards can be represented by *Cash* settlement.
  Allowable values: *Cash* or *Physical*. Defaults to *Physical* if left blank or omitted.
- SettlementData [Optional]: This node is used to specify the settlement of the cash flows on the value date.

A `SettlementData` node is shown in Listing 16, and the meanings and allowable values of its elements follow below.

- Currency: The currency in which the FX Forward is settled. This field is only used if settlement is *Cash*.
  Allowable values: See Table 15 `Currency`. Defaults to the sold currency if left blank or omitted.
- FXIndex: The FX reference index for determining the FX fixing at the value date. This field is required if settlement is *Cash* and the payment date is greater than the value date. Otherwise, it is ignored.
  Allowable values: The format of the `FXIndex` is "FX-FixingSource-CCY1-CCY2" as described in Table 21.
- Date [Optional]: If specified, this will be the payment date.
  Allowable values: See `Date` in Table 13. If left blank or omitted, defaults to the value date with some adjustments applied from the `Rules` sub-node.
- Rules [Optional]: If `Date` is left blank or omitted, this node will be used to derive the payment date from the value date. The `Rules` sub-node is shown in Listing 16, and the meanings and allowable values of its elements follow below.
  - PaymentLag [Optional]: The lag between the value date and the payment date.
    Allowable values: Any valid period, i.e. a non-negative whole number, optionally followed by D (days), W (weeks), M (months), Y (years). For cash settlement and if a FXIndex is specified defaults to the fx convention (field "SpotDays") if blank or omitted, otherwise to 0. If a whole number is given and no letter, it is assumed that it is a number of D (days).

– PaymentCalendar [Optional]: The calendar to be used when applying the payment lag.
Allowable values: See Table 17 `Calendar`. For cash settlement and if a FXIndex is specified defaults to the fx convention (field "AdvanceCalendar") if left blank or omitted, otherwise to NullCalendar (no holidays).

– PaymentConvention [Optional]: The roll convention to be used when applying the payment lag.
Allowable values: See Table 14 Roll Convention. For cash settlement and if a FXIndex is specified defaults to the fx convention ((field "Convention") if left blank or omitted, otherwise to Unadjusted.

Note that FX Forwards also cover Precious Metals forwards, i.e. with currencies XAU, XAG, XPT, XPD, and Cryptocurrency forwards, see supported Cryptocurrencies in Table 15.

Listing 16: Example ***`SettlementData`*** *node with* ***`Rules`*** *sub-node*

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

2.2.9 FX Average Forward

The `FXAverageForwardData` node is the trade data container for the *FxAverageForward* trade type. The structure with example values node is shown in Listing 17.

Listing 17: FX Average Forward data

```
<FxAverageForwardData>
    <PaymentDate>2023-04-09</PaymentDate>
    <!-- Schedule block that determines observation dates for FX averaging -->
    <ObservationDates>
       ...
    </ObservationDates>
    <FixedPayer>true</FixedPayer>
    <ReferenceNotional>8614</ReferenceNotional>
    <ReferenceCurrency>EUR</ReferenceCurrency>
    <SettlementNotional>10000</SettlementNotional>
    <SettlementCurrency>USD</SettlementCurrency>
    <FXIndex>FX-ECB-EUR-USD</FXIndex>
    <Settlement>Cash</Settlement>
</FxAverageForwardData>
```

The instrument's payoff is driven by an arithmetic average of observed FX rates,

expressed in terms of the node names:

$$\omega \times (\text{ReferenceNotional} \times \text{AverageFX} - \text{SettlementNotional})$$

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

- PaymentDate: The date of the settlement cash flow.
  Allowable values: See `Date` in Table 13.
- ObservationDates: Schedule data that determine the observation dates that are taken into account in the FX rate averaging. See section 2.3.4
- FixedPayer: If *true*, the payoff multiplier ω is set to 1, otherwise -1.
  Allowable values: *true*, *false*
- ReferenceNotional: The amount to be converted into settlement currency at the average FX rate
  Allowable values: Any positive real number.
- ReferenceCurrency: The currency of the reference notional above.
  Allowable values: See Table 15 `Currency`.
- SettlementNotional: The fixed amount to be paid or received depending on the fixed payer flag above
  Allowable values: Any positive real number.
- SettlementCurrency: The currency of the settlement notional above.
  Allowable values: See Table 15 `Currency`.
- FXIndex: The FX reference index for determining the FX fixing for averaging.
  Allowable values: The format of the `FXIndex` is "FX-FixingSource-CCY1-CCY2" as described in Table 21. Notice that since the payoff is based on an arithmetic average, the order of the currencies in the FX index matters: The averaging will be done on fx rates quoted as CCY1-CCY2 (foreign-domestic).

2.2.10 FX Swap

Payoff

An FX Swap is a contract that involves the exchange of a set amount of one currency for another at the start of the contract, and then the reverse exchange at the end of the contract. The exchange at the start uses the FX spot rate, whereas the exchange at the end uses the FX forward rate at the start.

Input

The `FXSwapData` node is the trade data container for the *FxSwap* trade type. The structure - including example values - of the `FXSwapData` node is shown in Listing 18. It contains no sub-nodes.

Listing 18: FX Swap data

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

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

- NearDate: The date of the initial fx exchange of the FX Swap.
  Allowable values: See `Date` in Table 13.
- NearBoughtCurrency: The currency to be bought in the initial exchange at near date, and sold in the final exchange at far date.
  Allowable values: See Table 15 `Currency`.
- NearBoughtAmount: The amount to be bought on near date.
  Allowable values: Any positive real number.
- NearSoldCurrency: The currency to be sold in the initial fx exchange at near date, and bought in the final exchange at far date.
  Allowable values: See Table 15 `Currency`.
- NearSoldAmount: The amount to be sold on near date.
  Allowable values: Any positive real number.
- FarDate: The date of the final fx exchange of the FX Swap.
  Allowable values: Any date further into the future than NearDate. See `Date` in Table 13.
- FarBoughtAmount: The amount to be bought on far date.
  Allowable values: Any positive real number.
- FarSoldAmount: The amount to be sold on far date.
  Allowable values: Any positive real number.
- Settlement [Optional]: Delivery type. Note that Non-Deliverable FX Swaps can be represented by *Cash* settlement, and that deliverable FX Swaps will be excluded from the CRIF output. Delivery type does not impact pricing in ORE.

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

Note that FX Swaps also cover Precious Metals swaps, i.e. with currencies XAU, XAG, XPT, XPD, and Cryptocurrency swaps, see supported Cryptocurrencies in Table 15.

2.2.11 FX Option

Payoff

A European FX option gives the buyer the right, but not the obligation, to exchange a set amount of one currency for another, at a predetermined exchange rate, at one predetermined time in the future. For this right the buyer pays a premium to the seller. Settlement can be either cash or physical delivery.

An American FX option gives the buyer the right, but not the obligation, to exchange a set amount of one currency for another, at a predetermined exchange rate, at any time during the life of the option up until the expiration date. The right to exchange of one currency for another can only be exercised once. For this right the buyer pays a premium to the seller.

- Settlement can be either cash or physical delivery.
- Payoff, i.e. the FX exchange or cash settlement, can take place at exercise or expiry.

Input

The `FXOptionData` node is the trade data container for the *FxOption* trade type. FX options with exercise styles *European* or *American* are supported. The `FXOptionData` node includes one and only one `OptionData` trade component sub-node plus elements specific to the FX Option. The structure of an `FXOptionData` node for an FX Option is shown in Listing 19.

Listing 19: FX Option data

```
<FxOptionData>
  <OptionData>
    <LongShort>Long</LongShort>
    <OptionType>Call</OptionType>
    <Style>European</Style>
    <Settlement>Cash</Settlement>
    <PayOffAtExpiry>false</PayOffAtExpiry>
    <ExerciseDates>
       <ExerciseDate>2026-03-01</ExerciseDate>
     </ExerciseDates>
     <Premiums>
       <Premium>
         <Amount>10900</Amount>
         <Currency>EUR</Currency>
         <PayDate>2020-03-01</PayDate>
       </Premium>
     </Premiums>
  </OptionData>
  <BoughtCurrency>EUR</BoughtCurrency>
  <BoughtAmount>1000000</BoughtAmount>
  <SoldCurrency>USD</SoldCurrency>
  <SoldAmount>1700000</SoldAmount>
</FxOptionData>
```

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

- OptionData: This is a trade component sub-node outlined in section 2.3.1. The relevant fields in the `OptionData` node for an FxOption are:

- `LongShort` The allowable values are *Long* or *Short*.
- `OptionType` The allowable values are *Call* or *Put*. For option type *Put*, Bought and Sold currencies/amounts are switched compared to the trade data node. For example, a holder of BoughtCurrency EUR SoldCurrency USD FX Call Option has the right to buy EUR using USD, while holder of the Put counterpart has the right to buy USD using EUR, or equivalently sell EUR for USD.
- `Style` The allowable values are *European* or *American*.
- `Settlement` The allowable values are *Cash* or *Physical*.
- `PayOffAtExpiry` [Optional] The allowable values are *true* for payoff at expiry, or *false* for payoff at exercise (relevant for *American* style FxOptions). Defaults to *true* if left blank or omitted.
- `AutomaticExercise` [Optional] The allowable values are *true* indicating Automatic Exercise is applicable and *false* indicates that it is not. Used if the FXOption expiry date is on the current date or in the past, and the payment date is in the future - so that there still is an outstanding cashflow if the FXOption was in the money on the expiry date. In this case, if AutomaticExercise is applied, the FX fixing on the expiry date is used to automatically determine the payoff and thus whether the option was exercised or not. Defaults to *false* if left blank or omitted.
- An `ExerciseDates` node where exactly one ExerciseDate date element must be given. For *American* style FxOptions the ExerciseDate represents the Expiry date, i.e. they can be exercised up until this date.
- A `PaymentData` [Optional] node can be added which defines the settlement date of the option payoff. See `PaymentData` in 2.3.1
- `Premiums` [Optional]: Option premium amounts paid by the option buyer to the option seller. See section 2.3.2

See 2.3.1 for further specifications of the `OptionData` node.

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

  Allowable values: See Table 15.

- BoughtAmount: The amount in the BoughtCurrency.

  Allowable values: Any positive real number.

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

  Allowable values: See Table 15.

- SoldAmount [Optional]: The amount in the SoldCurrency. Note that if Delta is omitted, the SoldAmount field is mandatory.

  Allowable values: Any positive real number.

- Delta [Optional]: The FX option delta. When a delta value is given the FX Option strike is derived from the delta, and the SoldAmount is ignored.

  Allowable values: Any non null real number. A SoldAmount or a Delta field is required, as the strike is derived from one or the other. Note: The delta to strike conversion is based on the valuation date. Therefore the strike will change day to day based on the market data variation. It is not possible to enter a seasoned trade with a Delta such that the trade strike (SoldAmount) is derived from the Delta on the trade date and then kept constant throughout the life of the trade.

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

  Allowable values: A valid FX index from the Table 21.

Note that FX Options also cover Precious Metals Options, i.e. with currencies XAU, XAG, XPT, XPD, and Cryptocurrency options, see supported Cryptocurrencies in Table 15.

2.2.12 FX Asian Option

Payoff

For an FX Asian option, the payoff is determined by the averaged foreign exchange rate over a pre-set period of time. At the expiration date, this product gives the buyer the right, but not the obligation, to obtain a cash amount of averaged rate in return for a predetermined strike rate. For this right the buyer pays a premium to the seller.

The payoff is

$$\text{Quantity} \times \max(\omega \cdot (A(0,T) - K), 0)$$

where:

- $A(0,T)$: the arithmetic average FX rate over the Asian observation period from start 0 to end T, expressed as amount of CCY2 per one unit of CCY1.
- K: strike FX rate, expressed as amount of CCY2 per one unit of CCY1.
- ω: 1 for a call option (ie receiving averaged FX and paying strike), -1 for a put option

Input

The `FxAsianOptionData` node is the trade data container for the *FxAsianOption* trade type. The `FxAsianOptionData` node includes one `OptionData` trade component sub-node plus elements specific to the FX Asian Option.

A FX Asian Option is a path-dependent option whose payoff depends upon the averaged foreign exchange rate over a pre-set period of time.

The structure of an example `FxAsianOptionData` node for a FX Asian Option is shown in Listing 20.

Listing 20: FX Asian Option data

```
<Trade id="FxAsianOption">
        <TradeType>FxAsianOption</TradeType>
        <Envelope>
                <CounterParty>CPTY_A</CounterParty>
                <NettingSetId>CPTY_A</NettingSetId>
                <AdditionalFields />
        </Envelope>
        <FxAsianOptionData>
                <Currency>USD</Currency>
                <Quantity>100</Quantity>
                <Strike>1.05</Strike>
                <Underlying>
                        <Type>FX</Type>
                        <Name>ECB-EUR-USD</Name>
                </Underlying>
                <OptionData>
                        <LongShort>Long</LongShort>
                        <OptionType>Call</OptionType>
                        <PayoffType>Asian</PayoffType>
                        <PayoffType2>Arithmetic</PayoffType2>
                        <ExerciseDates>
                                <ExerciseDate>2020-07-15</ExerciseDate>
                        </ExerciseDates>
                </OptionData>
                <Settlement>2020-07-20</Settlement>
                <ObservationDates>
                        <Rules>
                                <StartDate>2019-12-27</StartDate>
                                <EndDate>2020-07-06</EndDate>
                                <Tenor>1D</Tenor>
                                <Calendar>US</Calendar>
                                <Convention>F</Convention>
                                <TermConvention>F</TermConvention>
                                <Rule>Forward</Rule>
                        </Rules>
                </ObservationDates>
        </FxAsianOptionData>
</Trade>
```

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

- Currency: The payoff currency.
  Allowable values: See Table 15 `Currency`.
- Quantity: The quantity of the underlying currency (CCY1). See payoff formula above.
  Allowable values: all positive real numbers
- Strike: The strike of the option, expressed as amount of CCY2 per one unit of CCY1.
  Allowable values: all positive real numbers
- Underlying: An `Underlying` node where `Type` must be set to *FX* and `Name` is the

foreign exchange currency pair (on the form SOURCE-CCY1-CCY2) including the `Currency` above typically as CCY2 and another currency defined as the underlying currency as CCY1.
Allowable values: See 2.3.29

- OptionData: This is a trade component sub-node outlined in section 2.3.1. The relevant fields in the `OptionData` node for an FxAsianOption are:
  - `LongShort` The allowable values are *Long* or *Short*.
  - `OptionType` The allowable values are *Call* or *Put*.
  - `PayoffType` which must be set to *Asian* or *AverageStrike* to identify a fixed or floating strike asian payoff,
  - `PayoffType2` [Optional] can be optionally set to *Arithmetic* or *Geometric* and defaults to *Arithmetic* if not given.
  - An `ExerciseDates` node where exactly one ExerciseDate date element must be given.
  - A `PaymentData` [Optional] node can be added which defines the settlement of the option payoff.
  - A `Premiums` [Optional] node can be added to represent deterministic option premia to be paid by the option holder. See section 2.3.2
- Settlement[Optional]: The settlement date.
  Allowable values: See `Date` in Table 13. Defaults to the ExerciseDate if left blank or omitted.
- ObservationDates: The observation dates for the asian period, given as a rules-based or dates-based schedule, analogous to a `ScheduleData` node but called `ObservationDates`.
  Allowable values: See the definition in 2.3.4

2.2.13 FX Barrier Option

Payoff

An FX Barrier option is a path-dependent option whose existence depends upon an FX spot rate reaching a pre-set barrier level. Exercise is European.

This product has a continuously monitored single barrier with a Vanilla European FX Option Underlying.

Single FX Barrier options can be knock-in or knock-out:

- A knock-in option is a barrier option that only comes into existence/becomes active when the FX spot rate reaches the barrier level at any point in the option's life. Once a barrier is knocked-in, the option will not cease to exist until the option expires and effectively it becomes a Vanilla FX Option.
- A knock-out option starts its life active, but ceases to exist/becomes inactive, if the barrier is reached during the life of the option.

When a Single FX Barrier option expires inactive, the payoff may be zero, or there may be a cash rebate (barrier rebate) paid out as a fraction of the original option premium.

There are four main types of Single Barrier FX Options:

- Up-and-out: The FX spot price starts below the barrier level and has to move up for the option to be knocked out.
- Down-and-out: The FX spot price starts above the barrier level and has to move down for the option to become knocked out.
- Up-and-in: The FX spot price starts below the barrier level and has to move up for the option to become activated.
- Down-and-in: The FX spot price starts above the barrier level and has to move down for the option to become activated.

Input

The `FxBarrierOptionData` node is the trade data container for the *FxBarrierOption* trade type. The barrier level of an FX Barrier Option is quoted as the amount in SoldCurrency per unit BoughtCurrency. The `FxBarrierOptionData` node includes one `OptionData` trade component sub-node and one `BarrierData` trade component sub-node plus elements specific to the FX Barrier Option.

The structure of an example `FxBarrierOptionData` node for a FX Barrier Option is shown in Listing 21.

Listing 21: FX Barrier Option data

```
<FxBarrierOptionData>
    <OptionData>
        <LongShort>Long</LongShort>
        <!-- Bought and Sold currencies/amounts are switched for Put -->
        <OptionType>Call</OptionType>
        <Style>European</Style>
        <Settlement>Cash</Settlement>
        <ExerciseDates>
         <ExerciseDate>2021-12-14</ExerciseDate>
        </ExerciseDates>
        ...
    </OptionData>
    <BarrierData>
     <Type>UpAndIn</Type>
     <Levels>
        <Level>1.2</Level>
     </Levels>
     <Rebate>0.0</Rebate>
    </BarrierData>
    <StartDate>2019-01-25</StartDate>
    <Calendar>TARGET</Calendar>
    <FXIndex>FX-ECB-EUR-USD</FXIndex>
    <BoughtCurrency>EUR</BoughtCurrency>
    <BoughtAmount>1000000</BoughtAmount>
    <SoldCurrency>USD</SoldCurrency>
    <SoldAmount>1100000</SoldAmount>
</FxBarrierOptionData>
```

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

- `OptionData`: This is a trade component sub-node outlined in section 2.3.1. The relevant fields in the `OptionData` node for an FxBarrierOption are:
  - `LongShort` The allowable values are *Long* or *Short*.
  - `OptionType` The allowable values are *Call* or *Put*.
    Call means that the holder of the option, upon expiry - assuming knock-in or no knock-out - has the right to receive the BoughtAmount and pay the SoldAmount.
    Put means that the Bought and Sold currencies/amounts are switched compared to the trade data node. For example, holder of BoughtCurrency EUR SoldCurrency JPY FX Barrier Call Option has the right to buy EUR using JPY, while holder of the Put counterpart has the right to buy JPY using EUR, or equivalently sell EUR for JPY. An alternative to define the latter option is to copy the Call option with following changes:
    a) swapping BoughtCurrency with SoldCurrency, b) swapping BoughtAmount with SoldAmount and c) inverting the barrier level (for example changing 110 to 0.0090909). Here barrier level is quoted as amount of EUR per unit JPY, which is not commonly seen on market and inconsistent with the format in Call options. For these reasons, using Put/Call flag instead is recommended.
  - `Style` The FX Barrier Option type allows for *European* option exercise style only.
  - `Settlement` The allowable values are *Cash* or *Physical*.
  - A `PaymentData` [Optional] node can be added which defines the settlement of the option payoff.
  - An `ExerciseDates` node where exactly one ExerciseDate date element must be given.
  - `Premiums` [Optional]: Option premium amounts paid by the option buyer to the option seller. See section 2.3.2
- `BarrierData`: This is a trade component sub-node outlined in section 2.3.31. `Level` specified in BarrierData should be quoted as the amount in SoldCurrency per unit BoughtCurrency, with both currencies as defined in FxBarrierOptionData node. Note that the barrier `Level` stays quoted as SoldCurrency per unit BoughtCurrency, regardless of Put/Call.
- `StartDate` [Optional]: The start date for checking if a barrier has been breached prior to today's date. If omitted or left blank no check is made and it is assumed no barrier has been breached in the past. Has no impact if set to today's date or a date in the future.

  Allowable values: See `Date` in Table 13.
- `Calendar` [Optional]: The calendar associated with the FX Index. Required if StartDate is set to a date prior to today's date, otherwise optional.

Allowable values: See Table 17 Calendar.

- `FXIndex` [Optional]: A reference to an FX Index source to check if the barrier has been breached. Required if StartDate is set to a date prior to today's date, otherwise optional and can be omitted but not left blank.

  Allowable values: The format of the FX Index is"FX-SOURCE-CCY1-CCY2" as described in table 21.

- `FXIndexDailyLows` [Optional]: Refers to an FX Index that tracks the daily low quotes. This is used to check if the barrier was breached at any point during the day. If not provided, ORE will automatically derive the index name by appending the suffix *_LOW* to the FXIndex source (e.g. *FX-SOURCE_LOW-CCY1-CCY2*). If no fixings are available, the system will fall back to using the fixings from the FXIndex.

  Allowable values: The format of the FX Index is"FX-SOURCE-CCY1-CCY2" as described in table 21.

- `FXIndexDailyHighs` [Optional]: Refers to an FX Index that tracks the daily high quotes. This is used to check if the barrier was breached at any point during the day. If not provided, ORE will automatically derive the index name by appending the suffix *_HIGH* to the FXIndex source (e.g. *FX-SOURCE_HIGH-CCY1-CCY2*). If no fixings are available, the system will fall back to using the fixings from the FXIndex.

  Allowable values: The format of the FX Index is"FX-SOURCE-CCY1-CCY2" as described in table 21.

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

  Allowable values: See Table 15 `Currency`.

- `BoughtAmount`: The amount in the BoughtCurrency.

  Allowable values: Any positive real number.

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

  Allowable values: See Table 15 `Currency`.

- `SoldAmount`: The amount in the SoldCurrency.

  Allowable values: Any positive real number.

Note that FX Barrier Options also cover Precious Metals, i.e. with currencies XAU, XAG, XPT, XPD, and Cryptocurrencies, see supported Cryptocurrencies in Table 15.

2.2.14 FX Digital Barrier Option

Payoff

This product has a continuously monitored single barrier with an Cash-or-Nothing digital underlying option.

At expiry a digital FX Barrier option pays either the payoff of a digital Cash-or-Nothing underlying option depending upon an FX spot rate reaching a pre-set barrier level.

- Can be of the same four main types as a regular Barrier FX Option: Up-and-out, Down-and-out, Up-and-in, Down-and-in.
- A Knock-In or One-Touch option pays the underlying option payoff if the barrier is breached, and no payout otherwise.
- A Knock-Out or No-Touch option has no payout if the barrier is breached, and pays the underlying option payoff otherwise.
- The buyer of a Digital FX Barrier Option pays a premium to the seller.

Input

The `FxDigitalBarrierOptionData` node is the trade data container for the *FxDigitalBarrierOption* trade type. The `FxDigitalBarrierOptionData` node includes one `OptionData` trade component sub-node and one `BarrierData` trade component sub-node plus elements specific to the FX Digital Barrier Option.

The structure of an example `FxDigitalBarrierOptionData` node for a FX Digital Barrier Option is shown in Listing 22.

Listing 22: FX Digital Barrier Option data

```
<FxDigitalBarrierOptionData>
    <OptionData>
        <LongShort>Long</LongShort>
        <OptionType>Call</OptionType>
        <Style>European</Style>
        <ExerciseDates>
         <ExerciseDate>2021-12-14</ExerciseDate>
        </ExerciseDates>
        ...
    </OptionData>
    <BarrierData>
        <Type>DownAndIn</Type>
        <Levels>
            <Level>1.18</Level>
        </Levels>
    </BarrierData>
    <StartDate>2019-01-25</StartDate>
    <Calendar>TARGET</Calendar>
    <FXIndex>FX-ECB-EUR-USD</FXIndex>
    <Strike>1.1</Strike>
    <PayoffAmount>100000</PayoffAmount>
    <PayoffCurrency>USD</PayoffCurrency>
    <ForeignCurrency>EUR</ForeignCurrency>
    <DomesticCurrency>USD</DomesticCurrency>
</FxDigitalBarrierOptionData>
```

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

- OptionData: This is a trade component sub-node outlined in section 2.3.1. The relevant fields in the `OptionData` node for an FxDigitalBarrierOption are:
  - `LongShort` The allowable values are *Long* or *Short*.
  - `OptionType` The allowable values are *Call* or *Put*. Given knock-in or no knock-out, *Call* means that the digital payout will occur if the fx rate at the expiry date is above the given strike, and *Put* means that the digital payout will occur if the fx rate at the expiry date is below the given strike.
  - `Style` The FX Digital Barrier Option type allows for *European* option exercise style only.
  - An `ExerciseDates` node where exactly one ExerciseDate date element must be given.
  - `Premiums` [Optional]: Option premium amounts paid by the option buyer to the option seller.

    Allowable values: See section 2.3.2
- BarrierData: This is a trade component sub-node outlined in section 2.3.31.

  Note that the *FxDigitalBarrierOption* is a single barrier instrument, and can have only one BarrierData node with one barrier level.

  Level specified in BarrierData should be quoted as the amount in DomesticCurrency per one unit of ForeignCurrency, with both currencies as defined in FxDigitalBarrierOptionData node.

  Type specified in BarrierData can be one of: *UpAndIn, DownAndIn, UpAndOut, DownAndOut*
- StartDate[Optional]: The start date for checking if a barrier has been breached prior to today's date. If omitted or left blank no check is made and it is assumed no barrier has been breached in the past. Has no impact if set to today's date or a date in the future. If'StartDate' is provided then the fixings for dates between this date and the asof date are checked to see if the option was triggered. If no fixing is available then we skip that date. This is to allow for backwards compatibility.

  Allowable values: See `Date` in Table 13.
- Calendar[Optional]: The calendar associated with the FX Index. Required if StartDate is set to a date prior to today's date, otherwise optional.

  Allowable values: See Table 17 Calendar.
- FXIndex[Optional]: A reference to an FX Index source to check if the barrier has been breached. Required if StartDate is set to a date prior to today's date, otherwise optional, and can be omitted but not left blank.

  Allowable values: The format of the FX Index is"FX-SOURCE-CCY1-CCY2" as described in table 21.
- `FXIndexDailyLows` [Optional]: Refers to an FX Index that tracks the daily low quotes. This is used to check if the barrier was breached at any point during the

day. If not provided, ORE will automatically derive the index name by appending the suffix *_LOW* to the FXIndex source (e.g. *FX-SOURCE_LOW-CCY1-CCY2*). If no fixings are available, the system will fall back to using the fixings from the FXIndex.

Allowable values: The format of the FX Index is"FX-SOURCE-CCY1-CCY2" as described in table 21.

- `FXIndexDailyHighs` [Optional]: Refers to an FX Index that tracks the daily high quotes. This is used to check if the barrier was breached at any point during the day. If not provided, ORE will automatically derive the index name by appending the suffix *_HIGH* to the FXIndex source (e.g. *FX-SOURCE_HIGH-CCY1-CCY2*). If no fixings are available, the system will fall back to using the fixings from the FXIndex.

  Allowable values: The format of the FX Index is"FX-SOURCE-CCY1-CCY2" as described in table 21.

- Strike: The FX strike price, expressed as the amount in DomesticCurrency per one unit of ForeignCurrency.

  Allowable values: Any positive real number.

- PayoffAmount: The fixed payoff amount expressed in the PayoffCurrency. It is cash-or-nothing payoff that depends on the option being in or out of the money, and whether the barrier has been breached.

  Allowable values: Any positive real number.

- PayoffCurrency[Optional]: The payoff currency of the FX digital option is the currency of the payoff amount. Must be either the Domestic or Foreign currency for this trade, If omitted this defaults to DomesticCurrency as defined in FxDigitalBarrierOptionData node.

  Allowable values: See Table 15 `Currency`.

- ForeignCurrency: The foreign currency of the FX digital barrier option is equivalent to the bought currency.

  Allowable values: See Table 15 `Currency`.

- DomesticCurrency: The domestic currency of the FX digital barrier option is equivalent to the sold currency.

  Allowable values: See Table 15 `Currency`.

Note that FX Digital Barrier Options also cover Precious Metals, i.e. with currencies XAU, XAG, XPT, XPD, and Cryptocurrencies, see supported Cryptocurrencies in Table 15.

2.2.15 FX Digital Option

Payoff

A Digital FX Option is an option whose payout is either zero or a fixed predetermined amount (Cash-or-Nothing). Payout depends on whether the underlying FX spot rate

expires in-the-money at the expiration date.

- Digital FX Options have European exercise with payout at expiry.
- The buyer of a Digital FX Option pays a premium to the seller.

Input

The `FxDigitalOptionData` node is the trade data container for the *FxDigitalOption* trade type. The `FxDigitalOptionData` node includes one `OptionData` trade component sub-node plus elements specific to the FX Digital Option. The structure of an example `FxDigitalOptionData` node for a FX Digital Option is shown in Listing 23.

Listing 23: FX Digital Option data

```
<FxDigitalOptionData>
    <OptionData>
        <LongShort>Long</LongShort>
        <OptionType>Call</OptionType>
        <Style>European</Style>
        <ExerciseDates>
         <ExerciseDate>2021-12-14</ExerciseDate>
        </ExerciseDates>
        ...
    </OptionData>
    <Strike>1.1</Strike>
    <PayoffCurrency>USD</PayoffCurrency>
    <PayoffAmount>100000</PayoffAmount>
    <ForeignCurrency>EUR</ForeignCurrency>
    <DomesticCurrency>USD</DomesticCurrency>
</FxDigitalOptionData>
```

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

- OptionData: This is a trade component sub-node outlined in section 2.3.1. The relevant fields in the `OptionData` node for an FxDigitalOption are:
  - `LongShort` The allowable values are *Long* or *Short*.
  - `OptionType` The allowable values are *Call* or *Put*. *Call* means that the digital payout will occur if the fx rate at the expiry date is above the given strike, and *Put* means that the digital payout will occur if the fx rate at the expiry date is below the given strike.
  - `Style` The FX Digital Option type allows for *European* option exercise style only.
  - An `ExerciseDates` node where exactly one ExerciseDate date element must be given.
  - `Premiums` [Optional]: Option premium amounts paid by the option buyer to the option seller.

    Allowable values: See section 2.3.2

- Strike: The FX strike price, expressed as the amount in DomesticCurrency per one unit of ForeignCurrency.

  Allowable values: Any positive real number.

- PayoffCurrency[Optional]: The payoff currency of the FX digital option is the currency of the payoff amount. Must be either the Domestic or Foreign currency for this trade, If omitted this defaults to the domestic currency.

  Allowable values: See Table 15 `Currency`.

- PayoffAmount: The fixed payoff amount expressed in payoff currency. It is cash-or-nothing payoff that depends on the option being in or out of the money.

  Allowable values: Any positive real number.

- ForeignCurrency: The foreign currency of the FX digital option is equivalent to the bought currency.

  Allowable values: See Table 15 `Currency`.

- DomesticCurrency: The domestic currency of the FX digital option is equivalent to the sold currency.

  Allowable values: See Table 15 `Currency`.

Note that FX Digital Options also cover Precious Metals, i.e. with currencies XAU, XAG, XPT, XPD, and Cryptocurrencies, see supported Cryptocurrencies in Table 15.

2.2.16 FX Double Barrier Option

Payoff

An FX Double Barrier option is a path-dependent option whose existence depends upon an FX spot rate reaching one of the two pre-set barrier levels. Exercise is European.

This product has two continuously monitored barriers with a Vanilla European FX Option Underlying.

Double FX Barrier options can be knock-in or knock-out:

- A knock-in option is a barrier option that only comes into existence/becomes active when the FX spot rate reaches the one of the barrier level at any point in the option's life. Once a barrier is knocked-in, the option will not cease to exist until the option expires and effectively it becomes a Vanilla FX Option.

- A knock-out option starts its life active, but ceases to exist/becomes inactive, if the one of the barriers is reached during the life of the option.

When a Double FX Barrier option expires inactive, the payoff may be zero, or there may be a cash rebate (barrier rebate) paid out as a fraction of the original option premium.

Note that the current implementation does not support options with Knock-In-Knock-Out (KIKO) barrier or Knock-Out-Knock-In (KOKI) barriers. The former is covered by a separate FX KIKO Barrier Option instrument.

Input

The `FxDoubleBarrierOptionData` node is the trade data container for the *FxDoubleBarrierOption* trade type.

The barrier levels of an FX Double Barrier Option are quoted as the amount in SoldCurrency per unit BoughtCurrency. The `FxDoubleBarrierOptionData` node includes one `OptionData` trade component sub-node and one `BarrierData` trade component sub-node plus elements specific to the FX Double Barrier Option. The structure of an example `FxDoubleBarrierOptionData` node for a FX Double Barrier Option is shown in Listing 24.

Listing 24: FX Double Barrier Option data

```
<FxDoubleBarrierOptionData>
    <OptionData>
        <LongShort>Long</LongShort>
        <!-- Bought and Sold currencies/amounts are switched for Put -->
        <OptionType>Call</OptionType>
        <Style>European</Style>
        <ExerciseDates>
         <ExerciseDate>2021-12-14</ExerciseDate>
        </ExerciseDates>
        ...
    </OptionData>
    <BarrierData>
        <Type>KnockOut</Type> <!-- KnockOut or KnockIn -->
        <Levels>
            <Level>1.1</Level>
            <Level>1.2</Level>
        </Levels>
        <Rebate>0.0</Rebate>
    </BarrierData>
    <StartDate>2019-01-25</StartDate>
    <Calendar>TARGET</Calendar>
    <FXIndex>FX-ECB-EUR-USD</FXIndex>
    <BoughtCurrency>EUR</BoughtCurrency>
    <BoughtAmount>1000000</BoughtAmount>
    <SoldCurrency>USD</SoldCurrency>
    <SoldAmount>1100000</SoldAmount>
</FxDoubleBarrierOptionData>
```

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

- OptionData: This is a trade component sub-node outlined in section 2.3.1. The relevant fields in the `OptionData` node for an FxDoubleBarrierOption are:
  - `LongShort` The allowable values are *Long* or *Short*.
  - `OptionType` The allowable values are *Call* or *Put*.
    Call means that the holder of the option, upon expiry - assuming knock-in or no knock-out - has the right to receive the BoughtAmount and pay the SoldAmount.

Put means that the Bought and Sold currencies/amounts are switched compared to the trade data node. For example, holder of BoughtCurrency EUR SoldCurrency JPY FX Double Barrier Call Option has the right to buy EUR using JPY, while holder of the Put counterpart has the right to buy JPY using EUR, or equivalently sell EUR for JPY. An alternative to define the latter option is to copy the Call option with following changes: a) swapping BoughtCurrency with SoldCurrency, b) swapping BoughtAmount with SoldAmount and c) inverting the barrier level (for example changing 110 to 0.0090909). Here barrier level is quoted as amount of EUR per unit JPY, which is not commonly seen on market and inconsistent with the format in Call options. For these reasons, using Put/Call flag instead is recommended.

  - `Style` The FX Double Barrier Option type allows for *European* option exercise style only.
  - `Settlement` The allowable values are *Cash* or *Physical*.
  - A `PaymentData` [Optional] node can be added which defines the settlement of the option payoff.
  - An `ExerciseDates` node where exactly one ExerciseDate date element must be given.
  - `Premiums` [Optional]: Option premium amounts paid by the option buyer to the option seller.

    Allowable values: See section 2.3.2

- `BarrierData`: This is a trade component sub-node outlined in section 2.3.31. Levels specified in BarrierData should be quoted as the amount in SoldCurrnecy per unit BoughtCurrency, with both currencies as defined in FxDoubleBarrierOptionData node. Changing the option from Call to Put or vice versa does not require switching the barrier levels. Two levels in ascending order should be defined in `Levels`. `Type` should be *KnockOut* or *KnockIn*.
- `StartDate` [Optional]: The start date for checking if a barrier has been breached prior to today's date. If omitted or left blank no check is made and it is assumed no barrier has been breached in the past. Has no impact if set to today's date or a date in the future.

  Allowable values: See `Date` in Table 13.
- `Calendar` [Optional]: The calendar associated with the FX Index. Required if StartDate is set to a date prior to today's date, otherwise optional.

  Allowable values: See Table 17 Calendar.
- `FXIndex` [Optional]: A reference to an FX Index source to check if the barrier has been breached. Required if StartDate is set to a date prior to today's date, otherwise optional and can be omitted but not left blank.

  Allowable values: The format of the FX Index is"FX-SOURCE-CCY1-CCY2" as described in table 21.

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

  Allowable values: See Table 15 `Currency`.

- `BoughtAmount`: The amount in the BoughtCurrency.

  Allowable values: Any positive real number.

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

  Allowable values: See Table 15 `Currency`.

- `SoldAmount`: The amount in the SoldCurrency.

  Allowable values: Any positive real number.

2.2.17 FX Double Touch Option

Payoff

This product has two continuously monitored barriers with a Cash-or-Nothing digital underlying.

A FX Double Touch option pays either a fixed predetermined payoff amount or zero (Cash-or-Nothing) depending upon an FX spot rate reaching one of the pre-set barrier levels.

- Can be of the same two main types as a regular Barrier FX Option: Knock-In, Kncok-Out
- A Knock-In or Double One-Touch option has a fixed payout if one of the barriers is breached, and no payout otherwise.
- A Knock-Out or Double No-Touch option has no payout if one of the barriers is breached, and fixed payout otherwise.
- The buyer of a FX Double Touch Option pays a premium to the seller.

Note that the current implementation supports FX Double Touch options with payout at expiry only.

Input

The `FxDoubleTouchOptionData` node is the trade data container for the *FxDoubleTouchOption* trade type. The `FxDoubleTouchOptionData` node includes one `OptionData` trade component sub-node and one `BarrierData` trade component sub-node plus elements specific to the FX Double Touch Option.

The structure of an example `FxDoubleTouchOptionData` node for an FX Double Touch Option is shown in Listing 25.

Listing 25: FX Double Touch Option data

```
<FxDoubleTouchOptionData>
    <OptionData>
        <LongShort>Long</LongShort>
        <PayOffAtExpiry>true</PayOffAtExpiry>
        <ExerciseDates>
         <ExerciseDate>2021-12-14</ExerciseDate>
        </ExerciseDates>
        ...
    </OptionData>
    <BarrierData>
        ...
        <Type>KnockOut</Type> <!-- KnockOut or KnockIn -->
        <Levels>
            <Level>1.1</Level>
            <Level>1.2</Level>
        </Levels>
        ...
    </BarrierData>
    <ForeignCurrency>EUR</ForeignCurrency>
    <DomesticCurrency>USD</DomesticCurrency>
    <PayoffCurrency>USD</PayoffCurrency>
    <PayoffAmount>100000</PayoffAmount>
    <StartDate>2019-01-25</StartDate>
    <FXIndex>FX-ECB-EUR-USD</FXIndex>
    <Calendar>TARGET</Calendar>
</FxDoubleTouchOptionData>
```

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

- `OptionData`: This is a trade component sub-node outlined in section 2.3.1. The relevant fields in the `OptionData` node for an FxDoubleTouchOption are as below. Note that the `OptionType` can be omitted.
  - `LongShort` The allowable values are *Long* or *Short*.
  - `PayOffAtExpiry` [Optional] *true* for payoff at expiry and *false* for payoff at hit. Currently, for both *KnockOut* and *KnockIn* barriers, only payoff at expiry (i.e. *true*) is supported. Defaults to *true* if left blank or omitted.
  - An `ExerciseDates` node where exactly one ExerciseDate date element must be given.
  - `PaymentData` [Optional]: This defines the settlement of the option payoff.
  - `Premiums` [Optional]: Option premium amounts paid by the option buyer to the option seller.

    Allowable values: See section 2.3.2
- `BarrierData`: This is a trade component sub-node outlined in section 2.3.31. Two levels in ascending order should be defined in *Levels*. *Type* should be KnockOut or KnockIn. Levels specified in BarrierData should be quoted as the

amount in DomesticCurrency (sold currency) per unit ForeignCurrency (bought currency).

- `ForeignCurrency`: The foreign currency of the FX touch option is equivalent to the bought currency.

  Allowable values: See Table 15 `Currency`.

- `DomesticCurrency`: The domestic currency of the FX touch option is equivalent to the sold currency.

  Allowable values: See Table 15 `Currency`.

- `PayoffCurrency`: The payoff currency of the FX touch option is the currency of the payoff amount.

  Allowable values: See Table 15 `Currency`.

- `PayoffAmount`: The fixed payoff amount expressed in payoff currency. It is cash-or-nothing payoff that depends on the option being in or out of the money, and whether the barrier has been touched.

  Allowable values: Any positive real number.

- `StartDate` [Optional]: The start date for checking if a barrier has been breached prior to today's date. If omitted or left blank no check is made and it is assumed no barrier has been breached in the past. Has no impact if set to today's date or a date in the future.

  Allowable values: See `Date` in Table 13.

- `FXIndex` [Optional]: A reference to an FX Index source to check if the barrier has been breached. Required if StartDate is set to a date prior to today's date, otherwise optional, and can be omitted but not left blank.

  Allowable values: The format of the FX Index is"FX-SOURCE-CCY1-CCY2" as described in table 21.

- `Calendar` [Optional]: The calendar associated with the FX Index. Required if StartDate is set to a date prior to today's date, otherwise optional.

  Allowable values: See Table 17 Calendar.

2.2.18 FX European Barrier Option

Payoff

A FX European Barrier option gives the buyer the right, but not the obligation, to exchange a set amount of one currency for another, at a predetermined exchange rate, at one predetermined time in the future. This right may be withdrawn depending upon an FX spot rate reaching a predetermined barrier level at the predetermined time, the underlying is monitored only at expiry with a single barrier).

For this right the buyer pays a premium to the seller. Settlement can be either cash or physical delivery.

Single FX European Barrier options can be knock-in or knock-out:

- A knock-in option is a barrier option that only comes into existence/becomes active when the FX spot rate reaches the barrier level at expiry.
- A knock-out option starts its life active, but ceases to exist/becomes inactive, if the barrier is reached at expiry.

When a Single FX European Barrier option expires inactive, the payoff may be zero, or there may be a cash rebate (barrier rebate) paid out as a fraction of the original option premium.

There are four main types of Single European Barrier FX Options:

- Up-and-out: The FX spot price starts below the barrier level and has to move up for the option to be knocked out.
- Down-and-out: The FX spot price starts above the barrier level and has to move down for the option to become knocked out.
- Up-and-in: The FX spot price starts below the barrier level and has to move up for the option to become activated.
- Down-and-in: The FX spot price starts above the barrier level and has to move down for the option to become activated.

Input

The `FxEuropeanBarrierOptionData` node is the trade data container for the *FxEuropeanBarrierOption* trade type. The barrier level of an FX European Barrier Option is quoted as the amount in SoldCurrency per unit BoughtCurrency. The `FxEuropeanBarrierOptionData` node includes one `OptionData` trade component sub-node and one `BarrierData` trade component sub-node plus elements specific to the FX Barrier Option.

The structure of an example `FxEuropeanBarrierOptionData` node for a FX European Barrier Option is shown in Listing 26.

Listing 26: FX European Barrier Option data

```
<FxEuropeanBarrierOptionData>
    <OptionData>
        <LongShort>Long</LongShort>
        <!-- Bought and Sold currencies/amounts are switched for Put -->
        <OptionType>Call</OptionType>
        <Style>European</Style>
        <Settlement>Cash</Settlement>
        <ExerciseDates>
         <ExerciseDate>2021-12-14</ExerciseDate>
        </ExerciseDates>
        ...
    </OptionData>
    <BarrierData>
     <Type>UpAndIn</Type>
     <Levels>
        <Level>1.2</Level>
     </Levels>
     <Rebate>100000</Rebate>
    </BarrierData>
    <BoughtCurrency>EUR</BoughtCurrency>
    <BoughtAmount>1000000</BoughtAmount>
    <SoldCurrency>USD</SoldCurrency>
    <SoldAmount>1100000</SoldAmount>
</FxEuropeanBarrierOptionData>
```

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

- OptionData: This is a trade component sub-node outlined in section 2.3.1. The relevant fields in the `OptionData` node for an FxEuropeanBarrierOption are:
  - `LongShort` The allowable values are *Long* or *Short*.
  - `OptionType` The allowable values are *Call* or *Put*.
    Call means that the holder of the option, upon expiry - assuming knock-in or no knock-out - has the right to receive the BoughtAmount and pay the SoldAmount.
    Put means that the Bought and Sold currencies/amounts are switched compared to the trade data node. For example, holder of BoughtCurrency EUR SoldCurrency JPY FX European Barrier Call Option has the right to buy EUR using JPY, while holder of the Put counterpart has the right to buy JPY using EUR, or equivalently sell EUR for JPY. An alternative to define the latter option is to copy the Call option with following changes: a) swapping BoughtCurrency with SoldCurrency, b) swapping BoughtAmount with SoldAmount and c) inverting the barrier level (for example changing 110 to 0.0090909). Here barrier level is quoted as amount of EUR per unit JPY, which is not commonly seen on market and inconsistent with the format in Call options. For these reasons, using Put/Call flag instead is recommended.
  - `Style` The FX European Barrier Option type allows for *European* option exercise style only.

  - `Settlement` The allowable values are *Cash* or *Physical*.
  - An `ExerciseDates` node where exactly one ExerciseDate date element must be given.
  - A `PaymentData` [Optional] node can be added which defines the settlement date of the option payoff.
  - `Premiums` [Optional]: Option premium amounts paid by the option buyer to the option seller. See section 2.3.2

- BarrierData: This is a trade component sub-node outlined in section 2.3.31. Level specified in BarrierData should be quoted as the amount in SoldCurrency per unit BoughtCurrency, with both currencies as defined in FxEuropeanBarrierOptionData node. Changing the option from Call to Put or vice versa does not require switching the barrier level, i.e. the level stays quoted as SoldCurrency per unit BoughtCurrency, regardless of Put/Call.
- BoughtCurrency: The bought currency of the FX barrier option. See OptionData above for more details.

  Allowable values: See Table 15 `Currency`.
- BoughtAmount: The amount in the BoughtCurrency.

  Allowable values: Any positive real number.
- SoldCurrency: The sold currency of the FX barrier option. See OptionData above for more details.

  Allowable values: See Table 15 `Currency`.
- SoldAmount: The amount in the SoldCurrency.

  Allowable values: Any positive real number.

Note that FX European Barrier Options also cover Precious Metals, i.e. with currencies XAU, XAG, XPT, XPD, and Cryptocurrencies, see supported Cryptocurrencies in Table 15.

2.2.19 FX KIKO Barrier Option

Payoff

An FX KIKO Barrier option is an option with both a knock-out and a knock-in barrier. The knock-out barrier can happen at any time, and once the knock-in barrier is hit the trade becomes a single barrier knock-out trade. The KIKO option can only be exercised if the knock-out barrier is never touched and the knock-in barrier is touched at least once.

The barriers can be any of the four main barrier types:

- Up-and-out: The FX spot price starts below the barrier level and has to move up for the option to be knocked out.
- Down-and-out: The FX spot price starts above the barrier level and has to move down for the option to become knocked out.

- Up-and-in: The FX spot price starts below the barrier level and has to move up for the option to become activated.
- Down-and-in: The FX spot price starts above the barrier level and has to move down for the option to become activated.

However, one of the barriers must be a knock-in and the other a knock-out.

Input

The `FXKIKOBarrierOptionData` node is the trade data container for the *FxKIKOBarrierOption* trade type.

The `FXKIKOBarrierOptionData` node includes one `OptionData` trade component sub-node and two `BarrierData` trade component sub-nodes plus elements specific to the FX KIKO Barrier Option. The structure of an example `FXKIKOBarrierOptionData` node for a FX KIKO Barrier Option is shown in Listing 27.

Listing 27: FX KIKO Barrier Option data

```
<FxKIKOBarrierOptionData>
    <OptionData>
        <LongShort>Long</LongShort>
        <!-- Bought and Sold currencies/amounts are switched for Put -->
        <OptionType>Call</OptionType>
        <Style>European</Style>
        <Settlement>Cash</Settlement>
        <ExerciseDates>
         <ExerciseDate>2021-12-14</ExerciseDate>
        </ExerciseDates>
        ...
    </OptionData>
    <Barriers>
        <BarrierData>
            <Type>UpAndIn</Type>
            <Levels>
                <Level>1.2</Level>
            </Levels>
        </BarrierData>
        <BarrierData>
            <Type>DownAndOut</Type>
            <Levels>
                <Level>1.2</Level>
            </Levels>
        </BarrierData>
    </Barriers>
    <StartDate>2019-01-25</StartDate>
    <Calendar>TARGET</Calendar>
    <FXIndex>FX-ECB-EUR-USD</FXIndex>
    <BoughtCurrency>EUR</BoughtCurrency>
    <BoughtAmount>1000000</BoughtAmount>
    <SoldCurrency>USD</SoldCurrency>
    <SoldAmount>1100000</SoldAmount>
</FxKIKOBarrierOptionData>
```

The meanings and allowable values of the elements in the `FXKIKOBarrierOptionData`

node follow below.

- OptionData: This is a trade component sub-node outlined in section 2.3.1. The relevant fields in the `OptionData` node for an FxKIKOBarrierOption are:
  - `LongShort` The allowable values are *Long* or *Short*.
  - `OptionType` The allowable values are *Call* or *Put*.
    Call means that the holder of the option, upon expiry - assuming knock-in or no knock-out - has the right to receive the BoughtAmount and pay the SoldAmount.
    Put means that the Bought and Sold currencies/amounts are switched compared to the trade data node. For example, holder of BoughtCurrency EUR SoldCurrency JPY FX KIKO Barrier Call Option has the right to buy EUR using JPY, while holder of the Put counterpart has the right to buy JPY using EUR, or equivalently sell EUR for JPY. An alternative to define the latter option is to copy the Call option with following changes: a) swapping BoughtCurrency with SoldCurrency, b) swapping BoughtAmount with SoldAmount and c) inverting the barrier level (for example changing 110 to 0.0090909). Here barrier level is quoted as amount of EUR per unit JPY, which is not commonly seen on market and inconsistent with the format in Call options. For these reasons, using Put/Call flag instead is recommended.
  - `Style` The FX KIKO Barrier Option type allows for *European* option exercise style only.
  - `Settlement` The allowable values are *Cash* or *Physical*.
  - An `ExerciseDates` node where exactly one ExerciseDate date element must be given.
  - Premiums [Optional]: Option premium amounts paid by the option buyer to the option seller.

    Allowable values: See section 2.3.2
- Barriers: This node contains two barrierData nodes, one must be a KnockIn barrier (*UpAndIn* or *DownAndIn*) and the other must be a KnockOut barrier (*UpAndOut* or *DownAndOut*).
- BarrierData: This is a trade component sub-node outlined in section 2.3.31. FxKIKOBarrierOptions do not currently support rebates. Level specified in BarrierData should be quoted as the amount in SoldCurrency per unit BoughtCurrency, with both currencies as defined in FxKIKOBarrierOptionData node. Changing the option from Call to Put or vice versa does not require switching the barrier level, i.e. the level stays quoted as SoldCurrency per unit BoughtCurrency, regardless of Put/Call.
- StartDate[Optional]: The start date for checking if a barrier has been breached prior to today's date. If omitted or left blank no check is made and it is assumed no barrier has been breached in the past. Has no impact if set to today's date or a date in the future.

Allowable values: See `Date` in Table 13.

- Calendar[Optional]: The calendar associated with the FX Index. Required if StartDate is set to a date prior to today's date, otherwise optional.

  Allowable values: See Table 17 Calendar.

- FXIndex[Optional]: A reference to an FX Index source to check if the barrier has been breached. Required if StartDate is set to a date prior to today's date, otherwise optional and can be omitted but not left blank.

  Allowable values: The format of the FX Index is"FX-SOURCE-CCY1-CCY2" as described in table 21.

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

  Allowable values: See Table 15 `Currency`.

- BoughtAmount: The amount in the BoughtCurrency.

  Allowable values: Any positive real number.

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

  Allowable values: See Table 15 `Currency`.

- SoldAmount: The amount in the SoldCurrency.

  Allowable values: Any positive real number.

Note that FX KIKO Options also cover Precious Metals, i.e. with currencies XAU, XAG, XPT, XPD, and Cryptocurrencies, see supported Cryptocurrencies in Table 15.

2.2.20 FX Touch Option

Payoff

This product has a continuously monitored single barrier with a Cash-or-Nothing digital underlying.

At expiry a FX Touch Option pays either a fixed predetermined payoff amount or zero (Cash-or-Nothing) depending upon an FX spot rate reaching a pre-set barrier level.

- Can be of the same four main types as a regular Barrier FX Option: Up-and-out, Down-and-out, Up-and-in, Down-and-in.
- A Knock-In or One-Touch option has a fixed payout if the barrier is breached, and no payout otherwise. The payout can be at expiry or at hit.
- A Knock-Out or No-Touch option has no payout if the barrier is breached, and fixed payout at expiry otherwise.
- The buyer of a FX Touch Option pays a premium to the seller.

Input

The `FxTouchOptionData` node is the trade data container for the *FxTouchOption* trade type. The `FxTouchOptionData` node includes one `OptionData` trade component sub-node and one `BarrierData` trade component sub-node plus elements specific to the FX Touch Option.

The structure of an example `FxTouchOptionData` node for an FX Touch Option is shown in Listing 28.

Listing 28: FX Touch Option data

```
<FxTouchOptionData>
    <OptionData>
        <LongShort>Long</LongShort>
        <PayOffAtExpiry>true</PayOffAtExpiry>
        <ExerciseDates>
         <ExerciseDate>2021-12-14</ExerciseDate>
        </ExerciseDates>
        ...
    </OptionData>
    <BarrierData>
     <Type>DownAndOut</Type>
     <Levels>
      <Level>0.009</Level>
     </Levels>
    </BarrierData>
    <ForeignCurrency>JPY</ForeignCurrency>
    <DomesticCurrency>USD</DomesticCurrency>
    <PayoffCurrency>USD</PayoffCurrency>
    <PayoffAmount>100000</PayoffAmount>
    <StartDate>2019-01-25</StartDate>
    <FXIndex>FX-TR20H-USD-JPY</FXIndex>
    <Calendar>NYB,TKB</Calendar>
</FxTouchOptionData>
```

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

- OptionData: This is a trade component sub-node outlined in section 2.3.1. The `OptionType` sub-node is not required and is inferred from the `BarrierData` type (i.e. *Call* for an Up barrier, and *Put* for a Down barrier). The relevant fields in the `OptionData` node for an FxTouchOption are:
  - `LongShort` The allowable values are *Long* or *Short*.
  - `PayOffAtExpiry` [Optional] *true* for payoff at expiry and *false* for payoff at hit. For UpAndOut and DownAndOut barrier, only payoff at expiry (*true*) is supported. Defaults to *true* if left blank or omitted. This field is ignored in pricing, and the option payoff will be calculated at expiry. This field only has an impact on the description of the trade economics. The *GenericBarrierOption* can also be used to 'replicate' the *FXTouchOption* with payoff at hit if required.
  - An `ExerciseDates` node where exactly one `ExerciseDate` date element must be given.

- A `PaymentData` [Optional] node can be added which defines the settlement of the option payoff. If the option is payoff at hit, (i.e. `PayoffAtExpiry` is *false*), the option payment data must be rules-based, and the `RelativeTo` sub-node of (`Rules`) must be set to *Exercise*.
- `Premiums` [Optional]: Option premium amounts paid by the option buyer to the option seller.

  Allowable values: See section 2.3.2

- `BarrierData`: This is a trade component sub-node outlined in section 2.3.31. Level specified in BarrierData should be quoted as the amount in DomesticCurrency (sold currency) per unit ForeignCurrency (bought currency). Note that the level stays quoted as DomesticCurrency per unit ForeignCurrency, regardless of barrier type.
- `ForeignCurrency`: The foreign (bought) currency of the FX touch option.

  Allowable values: See Table 15 `Currency`.
- `DomesticCurrency`: The domestic (sold) currency of the FX touch option.

  Allowable values: See Table 15 `Currency`.
- `PayoffCurrency`: The payoff currency of the FX touch option is the currency of the payoff amount.

  Allowable values: See Table 15 `Currency`.
- `PayoffAmount`: The fixed payoff amount expressed in payoff currency. It is cash-or-nothing payoff that depends on the option being in or out of the money, and whether the barrier has been touched.

  Allowable values: Any positive real number.
- `StartDate` [Optional]: The start date for checking if a barrier has been breached prior to today's date. If omitted or left blank no check is made and it is assumed no barrier has been breached in the past. Has no impact if set to today's date or a date in the future.

  Allowable values: See `Date` in Table 13.
- `FXIndex` [Optional]: A reference to an FX Index source to check if the barrier has been breached. Required if StartDate is set to a date prior to today's date, otherwise optional, and can be omitted but not left blank.

  Allowable values: The format of the FX Index is"FX-SOURCE-CCY1-CCY2" as described in table 21.
- `FXIndexDailyLows` [Optional]: Refers to an FX Index that tracks the daily low quotes. This is used to check if the barrier was breached at any point during the day. If not provided, ORE will automatically derive the index name by appending the suffix *_LOW* to the FXIndex source (e.g. *FX-SOURCE_LOW-CCY1-CCY2*). If no fixings are available, the system will fall back to using the fixings from the FXIndex.

Allowable values: The format of the FX Index is"FX-SOURCE-CCY1-CCY2" as described in table 21.

- `FXIndexDailyHighs` [Optional]: Refers to an FX Index that tracks the daily high quotes. This is used to check if the barrier was breached at any point during the day. If not provided, ORE will automatically derive the index name by appending the suffix *_HIGH* to the FXIndex source (e.g. *FX-SOURCE_HIGH-CCY1-CCY2*). If no fixings are available, the system will fall back to using the fixings from the FXIndex.

  Allowable values: The format of the FX Index is"FX-SOURCE-CCY1-CCY2" as described in table 21.

- `Calendar` [Optional]: The calendar associated with the FX Index. Required if StartDate is set to a date prior to today's date, otherwise optional.

  Allowable values: See Table 17 Calendar.

Note that FX Touch Options also cover Precious Metals, i.e. with currencies XAU, XAG, XPT, XPD, and Cryptocurrencies, see supported Cryptocurrencies in Table 15.

2.2.21 FX Variance and Volatility Swap

Payoff

An **FX Variance Swap** has a payoff, similar to an Equity Variance Swap's payoff 2.2.34, that depends on the volatility of an underlying FX rate. The swap counterparties agree to exchange a pre-agreed variance level (the strike) for the actual amount of variance realized over an observation period.

The strike is typically set at ATM so the swap initially has zero value. If the subsequent realized volatility is above the strike level, the buyer of a variance swap who is long volatility will have a positive NPV, and the seller who is short volatility, will have a negative NPV.

Payoff:

$$N \cdot \left(\text{RealisedVol}^2 - K^2\right)$$

where

- N: Notional, also called Variance Notional, determined as Vega Notional/$(2K)$.
- Vega Notional: Notional in terms of volatility units.
- RealisedVol:

$$\sqrt{252 \cdot \sum_{t=1}^{N} \frac{1}{\text{TradingDays}} \left(\ln \frac{P_t}{P_{t-1}}\right)^2} \cdot 100$$

- TradingDays: the number of days which are expected to be scheduled trading days in the observation period
- P_0: the official closing of the underlying at the observation start date
- P_t: the official closing of the underlying at any observation date t, or at observation end date

- K: the strike volatility

An **FX Volatility Swap** is closely related to the FX Variance Swap with payoff

$$N \cdot (\text{RealisedVol} - K)\,.$$

Input

The `FxVarianceSwapData` node is the trade data container for the *FxVarianceSwap* trade type. Only vanilla variance swaps are supported by this trade type - exotic variance swaps are supported by ScriptedTrade. The structure of an example `VarianceSwapData` node for an FX variance swap is shown in Listing 29.

Listing 29: Variance Swap data

```
<FxVarianceSwapData>
        <StartDate>2018-05-10</StartDate>
        <EndDate>2018-11-12</EndDate>
        <Currency>EUR</Currency>
        <Underlying>
          <Type>FX</Type>
          <Name>ECB-EUR-JPY</Name>
        </Underlying>
        <LongShort>Long</LongShort>
        <Strike>0.05</Strike>
        <Notional>200000</Notional>
        <Calendar>EUR</Calendar>
        <MomentType>Variance</MomentType>
</FxVarianceSwapData>
```

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

- StartDate: The variance swap start date.
  Allowable values: See `Date` in Table 13.
- EndDate: The variance swap end date.
  Allowable values: See `Date` in Table 13.
- Currency: The bought currency of the variance swap.
  Allowable values: See Table 15 `Currency`.
- Name: The identifier of the underlying currency pair.
  Allowable values: A string of the form SOURCE-CCY1-CCY2, where SOURCE is the fixing source and the fixing is expressed as amount in CCY2 per one unit of CCY1.
  See Table 21. Note that FxVarianceSwap is an exception in that the ordering of CCY1 and CCY2 must be set up as for `FxIndex`.
- Underlying: This node may be used as an alternative to the `Name` node to specify the underlying FX. The `Underlying` node is described in further detail in Section 2.3.29.

- LongShort: Defines whether the trade is long in the FX variance. For the avoidance of doubt, a long FX swap has positive value if the realised variance exceeds the variance strike.
  Allowable values: *Long, Short*
- Strike: The volatility strike K_{vol} of the variance swap quoted absolutely (i.e. not as a percent). If the swap was struck in terms of variance, the square root of that variance should be used here.
  Allowable values: Any positive real number.
- Notional: The vega notional of the variance swap. This is the notional in terms of volatility units (like the strike). If the swap was struck in terms of a variance notional N_{var}, the corresponding vega notional is given by $N_{vol} = N_{var} * 2 * 100 * K_{vol}$ (where K_{vol} is in absolute terms).
  Allowable values: Any non-negative real number.
- Calendar: The calendar determining the observation/fixing dates according to which variance is accrued is the combination of the calendar(s) given here plus the combined calendars of the two involved currencies.
  Allowable values: See Table 17.
- MomentType[Optional]: A flag to distinguish if the swap is struck in terms of volatility or variance. The MomentType should be set to *Volatility* or *Variance* depending on the payoff. Note that MomentType does not necessarily need to be equivalent to the way the Strike is quoted which is always as a Volatility.
  Allowable values: *Volatility* or *Variance*. Defaults to *Variance* if left blank or omitted.

Note that FX Variance and Volatility Swaps also cover Precious Metals, i.e. with currencies XAU, XAG, XPT, XPD, and Cryptocurrencies, see supported Cryptocurrencies in Table 15.

2.2.22 Equity Option

Payoff

A **European Equity Option** gives the buyer the right, but not the obligation, to buy a set number of shares of a single name equity or an equity index, at a predetermined strike price, at the end of the contract. For this right the buyer pays a premium to the seller.

An **American Equity Option** gives the buyer the right to buy at any time during the life of the option up until the expiration date. The right to buy the shares can only be exercised once.

A **Quanto European Equity Option** is a European Equity Option where the currency that the underlying equity (or equity index) is quoted in and the option payoff currency are different. The implied FX rate between the underlying currency and payoff currency at the option settlement date is then equal to one.

In a **European Equity Composite Option** the strike currency is different from the underlying currency. This is unrelated to the *CompositeTrade* trade type. For

example, the composite call option payoff at expiry is

$$\max(X(t) * S(t) - K, 0.0)$$

where $X(t)$ is the exchange rate at expiry that converts from underlying currency tinto strike currency.

- Settlement can be either cash or physical delivery.
- Payoff, i.e. the cash-equity exchange or cash settlement, can take place at exercise or expiry.

Input

The `EquityOptionData` node is the trade data container for the *EquityOption* trade type. Equity options with exercise styles *European* and *American* are supported.

The `EquityOptionData` node includes one and only one `OptionData` trade component sub-node plus elements specific to the equity option. The structure of an example `EquityOptionData` node for an equity option is shown in Listing 30.

Listing 30: Equity Option data

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

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

- OptionData: This is a trade component sub-node outlined in section 2.3.1 Option Data. The relevant fields in the `OptionData` node for an EquityOption are:
  - `LongShort`: The allowable values are *Long* or *Short*.
  - `OptionType`: The allowable values are *Call* or *Put*. *Call* means that the option holder has the right to buy the given quantity of the underlying equity at the strike price. *Put* means that the option holder has the right to sell the given quantity of the underlying equity at the strike price.
  - `Style`: The allowable values are *European* and *American*.

- `Settlement`: The allowable values are *Cash* or *Physical*. If `Currency` and underlying equity currency are different, i.e. Quanto payoff, this must be set to *Cash*.
- `PayOffAtExpiry` [Optional]: The allowable values are *true* for payoff at expiry, or *false* for payoff at exercise. This field is relevant for *American* style EquityOptions, and defaults to *true* if left blank or omitted.
- An `ExerciseDates` node where exactly one ExerciseDate date element must be given.
- `PaymentData` [Optional]: Node used to set the payment date if it differs from the exercise date. Note that for quanto and compo EquityOptions the payment date cannot differ from the exercise date.
- `Premiums` [Optional]: Node for Option premium amounts paid by the option buyer to the option seller.

  Allowable values: See section 2.3.2

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

  Allowable values: See `Name` for equity trades in Table 24.

- Underlying: This node may be used as an alternative to the `Name` node to specify the underlying equity. This in turn defines the equity curve used for pricing. The `Underlying` node is described in further detail in Section 2.3.29.

- Currency: The payment currency of the equity option.

  Allowable values: See `Currency` and `Minor Currencies` in Table 13. If this is different to the currency that the underlying equity is quoted in, then a Quanto payoff will be applied. Using the corresponding major currency for an equity quoted in the minor currency will not correspond to a Quanto payoff.

- Strike[Mandatory except if StrikeData node is used]: The option strike price.

  Allowable values: Any positive real number.

- StrikeCurrency [Mandatory for Quanto/Compo, Optional otherwise]: The currency that the `Strike` is quoted in. If the option is Quanto, then this field must not be left blank, and must equal the currency that the underlying equity is quoted in, up to the minor/major currency. For example, if the underlying equity is quoted in GBP, then`StrikeCurrency` must be either *GBP* or *GBp*. If the option is a Compo option, then this field must not be left blank, and it must equal the payment currency of the option and different to the underlying currency.

  Note:
  Quanto: Payment currency and the currency the underlying equity is quoted in differ. StrikeCurrency is in the currency the equity is quoted in.
  Compo (Composite): Payment currency and the currency the underlying equity is quoted in differ. StrikeCurrency is in the payment currency.

  Allowable values: See Fiat Currencies and Minor Currencies in Table 15. Must be the major or minor currency of the `Currency` field above, or in the Quanto

case it must be the major or minor currency the underlying is quoted in. If left blank or omitted, and payment currency is the same as the equity currency, it defaults to the `Currency` field (payment currency) above.

- StrikeData[Optional]: Alternatively, instead of the `Strike` and the `StrikeCurrency` fields above a `StrikeData` node can be used as described in Section 2.3.30. Note that for EquityOptions only `StrikePrice` is supported within the `StrikeData` node, and not `StrikeYield`.
- Quantity: The number of units of the underlying covered by the transaction.

  Allowable values: Any positive real number.

2.2.23 Equity Futures Option

The `EquityFutureOptionData` node is the trade data container for the *EquityFutureOption* trade type. Equity options with exercise styles *European* and *American* are supported. The `EquityFutureOptionData` node includes one and only one `OptionData` trade component sub-node plus elements specific to the equity future option. The structure of an example `EquityFutureOptionData` node for an equity option is shown in Listing 31.

Listing 31: Equity Future Option data

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

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

- OptionData: This is a trade component sub-node outlined in section 2.3.1 Option Data. The relevant fields in the `OptionData` node for an EquityOption are:
  - `LongShort` The allowable values are *Long* or *Short*.

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

  Allowable values: See section 2.3.2

- Name: The identifier of the underlying equity or equity index.
  Allowable values: See `Name` for equity trades in Table 23.
- Underlying: This node may be used as an alternative to the `Name` node to specify the underlying equity. This in turn defines the equity curve used for pricing. The `Underlying` node is described in further detail in Section 2.3.29.
- Currency: The currency of the equity option.
  Allowable values: See Table 15.
- StrikeData: The option strike price.
  Allowable values: Only supports `StrikePrice` as described in Section 2.3.30.
- Quantity: The number of units of the underlying covered by the transaction.
  Allowable values: Any positive real number.
- FutureExpiryDate [Optional]: If `IsFuturePrice` is `true` and the underlying is a future contract settlement price, this node allows the user to specify the expiry date of the underlying future contract.

  Allowable values: This should be a valid date as outlined in Table 13. If not provided, it is assumed that the future contract's expiry date is equal to the option expiry date provided in the `OptionData` node.

2.2.24 Equity Forward

Payoff

An Equity Forward contract is an agreement between two counterparties to buy/sell a set number of shares of a single name equity or an equity index, at a predetermined strike price, at the end of the contract. An equity forward does not involve any upfront payment, does not pay dividends, and settlement is cash.

Input

The `EquityForwardData` node is the trade data container for the *EquityForward* trade type. Vanilla equity forwards are supported. The structure of an example `EquityForwardData` node for an equity forward is shown in Listing 32.

Listing 32: Equity Forward data

```
<EquityForwardData>
  <LongShort>Long</LongShort>
  <Maturity>2018-06-30</Maturity>
  <Name>RIC:.SPX</Name>
  <Currency>USD</Currency>
  <Strike>2147.56</Strike>
  <StrikeCurrency>USD</StrikeCurrency>
  <Quantity>17000</Quantity>
</EquityForwardData>
```

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

- LongShort: Defines whether the underlying equity will be bought (long) or sold (short).
  Allowable values: *Long*, *Short*.
- Maturity: The maturity date of the forward contract, i.e. the date when the underlying equity will be bought/sold.
  Allowable values: Any date string, see `Date` in Table 13.
- Name: The identifier of the underlying equity or equity index. Allowable values: See `Name` for equity trades in Table 23.
- Underlying: This node may be used as an alternative to the `Name` node to specify the underlying equity. This in turn defines the equity curve used for pricing. The `Underlying` node is described in further detail in Section 2.3.29.
- Currency: The payment currency of the equity forward. If the equity underlying is quoted in a different currency, a `FXIndex` in the `SettlementData` sub-node is required to convert the payoff into the payment currency.
  Allowable values: See Fiat Currencies and Minor Currencies in Table 15.
- Strike: The agreed buy/sell price of the equity forward.
  Allowable values: Any positive real number.
- StrikeCurrency: [Optional] The currency of the strike value. The strike value has to be in underlying quotation currency. If the strike currency is quoted in the minor currency, the strike value will be converted to the major currency. Defaults to the payment currency if omitted or blank.
  Allowable values: See Fiat Currencies and Minor Currencies in Table 15.
- Quantity: The number of units of the underlying equity to be bought/sold.
  Allowable values: Any positive real number.
- SettlementData [Optional]: This node is used to specify the settlement of the cash flows.

The strike value must be quoted in the same currency as the underlying. The underlying prices are always converted to the major underlying currency during curve building. If the strike is quoted in the minor underlying currency, it will be also converted to the major underlying currency. If the strike currency is blank or omitted, it defaults to payment currency, in this case the payment currency needs to be the same as the underlying currency and same logic applies for minor to major currency conversion.

A `SettlementData` node is shown in Listing 33, and the meanings and allowable values of its elements follow below.

- FXIndex: The FX reference index for determining the FX fixing used to convert the amount from the underlying equity quotation currency to the payment currency. This field is required if the underlying currency doesn't match the deal currency. Otherwise, it is ignored.
  Allowable values: The format of the `FXIndex` is "FX-FixingSource-CCY1-CCY2" as described in Table 21.

- Date [Optional]: If specified, this will be the payment date.
  Allowable values: See `Date` in Table 13. If left blank or omitted, the payment date will be derived from the maturity date applying the `PaymentLag`, `PaymentCalendar` and the `PaymentConvention` as defined in the `Rules` sub-node.

- Rules [Optional]: If `Date` is left blank or omitted, this node will be used to derive the payment date from the maturity date. The `Rules` sub-node is shown in Listing 33, and the meanings and allowable values of its elements follow below.

  - PaymentLag [Optional]: The lag between the maturity date and the payment date.
    Allowable values: Any valid period, i.e. a non-negative whole number, optionally followed by D (days), W (weeks), M (months), Y (years). Defaults to 0. If a whole number is given and no letter, it is assumed that it is a number of D (days).

  - PaymentCalendar [Optional]: The calendar to be used when applying the payment lag.
    Allowable values: See Table 17 `Calendar`. Defaults to to NullCalendar (no holidays) if left blank or omitted.

  - PaymentConvention [Optional]: The roll convention to be used when applying the payment lag.
    Allowable values: See Table 14 Roll Convention. Defaults to Unadjusted if left blank or omitted.

Listing 33: Example `SettlementData` *node with* `Rules` *sub-node*

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

2.2.25 Equity Swap

Payoff

An equity swap is a swap where one of the legs has a floating rate with coupon payments linked to an equity price, either a single stock or equity index.

- The non equity leg is either a fixed rate or a floating rate based off an IBOR index.
- The notional amount for the equity leg can be either fixed, or resettable at specified dates - this equates to a fixed quantity of shares in the underlying equity.
- There are two types of equity coupons, Price Return and Total Return. Price Return coupons pay the difference in equity price between the coupon start and end dates, while a Total Return coupon also includes dividend payments during the period.
- The equity price can optionally be converted from the equity ccy to the equity leg ccy using the FX spot rate observed at the equity fixing dates.

Input

An Equity Swap uses its own trade type *EquitySwap*, and is set up using a `EquitySwapData` node with one leg of type *Equity* and one more leg - called Funding leg - that can be either *Fixed* or *Floating*. Listing 34 shows an example. The Equity leg contains an additional `EquityLegData` block. See 2.3.16 for details on the Equity leg specification.

Note that the *Equity* leg of an *EquitySwap* can only include one single underlying equity name (that can be an equity index name). For instruments with more than one underlying equity name, TradeType *TotalReturnSwap* (GenericTRS) should be used instead.

Cross currency *EquitySwaps* are supported, i.e. the Equity and the Funding legs do not need to have the same currency. However, if the Funding leg uses `Indexings` with `FromAssetLeg` set to *true* to derive the notionals from the Equity leg, then the Funding leg must use the same currency as the Equity leg.

Note that pricing for an *EquitySwap* is based on discounted cashflows, whereas pricing for a *TotalReturnSwap* (GenericTRS) on an equity underlying uses the accrual

method. The accrual method is common practice when daily unwind rights are present in the trade terms.

Also note that, unlike other leg types, the `DayCounter` field is optional for an *Equity* leg, and defaults to *ACT/365* if left blank or omitted. The daycount convention for the equity leg of an equity swap does not impact pricing, only the accrued amount (displayed in cashflows).

Listing 34: Equity Swap Data

```
<EquitySwapData>
  <LegData>
    <LegType>Floating</LegType>
    <Payer>true</Payer>
    <DayCounter>ACT/365</DayCounter>
    ...
  </LegData>
  <LegData>
    <LegType>Equity</LegType>
    <Payer>false</Payer>
    <DayCounter>ACT/365</DayCounter>
    ...
    <EquityLegData>
    ...
    </EquityLegData>
  </LegData>
</EquitySwapData>
```

If the equity swap has a resetting notional, typically the Funding leg's notional will be aligned with the equity leg's notional. To achieve this, `Indexings` on the floating leg can be used, see 2.3.8. In the context of equity swaps the indexings can be defined in a simplified way by adding an `Indexings` node with a subnode `FromAssetLeg` set to *true* to the Funding leg's `LegData` node. The `Notionals` node is not required in the Funding leg's LegData in this case. An example is shown in listing 35.

Listing 35: Equity Swap Data with notional reset and FX indexing

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

2.2.26 Dividend Swap

An Dividend Swap uses its the trade type *EquitySwap*, shown above 2.2.25, and is set up using a `EquitySwapData` node with one leg of type *Equity*, with *ReturnType* equal to *Dividend* and one more leg that can be either *Fixed* or *Floating*. Listing 36 shows an example.

An example is shown in listing 35.

Listing 36: Dividend Swap Data

```
<EquitySwapData>
    <LegData>
        ...
    </LegData>
    <LegData>
        <Payer>false</Payer>
        <LegType>Equity</LegType>
        <Currency>EUR</Currency>
        <PaymentConvention>Following</PaymentConvention>
        <DayCounter>A360</DayCounter>
        <EquityLegData>
            <ReturnType>Dividend</ReturnType>
            <Underlying>
             <Type>Equity</Type>
             <Name>.STOXX50E</Name>
             <IdentifierType>RIC</IdentifierType>
            </Underlying>
            <Quantity>10000</Quantity>
        </EquityLegData>
        <ScheduleData>
            <Rules>
                <StartDate>2018-12-31</StartDate>
                <EndDate>2020-12-31</EndDate>
                <Tenor>6M</Tenor>
                <Calendar>EUR</Calendar>
                <Convention>ModifiedFollowing</Convention>
                <Rule>Forward</Rule>
            </Rules>
        </ScheduleData>
    </LegData>
</EquitySwapData>
```

2.2.27 Equity Asian Option

An Equity Asian Option is a path-dependent option whose payoff depends upon the averaged price of an Equity underlying over a pre-set period of time.

The `EquityAsianOptionData` node is the trade data container for the *EquityAsianOption* trade type. The `EquityAsianOptionData` node includes one `OptionData` trade component sub-node plus elements specific to the Equity Asian Option.

The structure of an example `EquityAsianOptionData` node for an Equity Asian Option is shown in Listing 37.

Listing 37: Equity Asian Option data

```
<Trade id="EquityAsianOption">
        <TradeType>EquityAsianOption</TradeType>
        <Envelope>
                <CounterParty>CPTY_A</CounterParty>
                <NettingSetId>CPTY_A</NettingSetId>
                <AdditionalFields />
        </Envelope>
        <EquityAsianOptionData>
                <Quantity>100</Quantity>
                <Currency>USD</Currency>
                <StrikeData>
                        <Value>3100</Value>
                        <Currency>USD</Currency>
                </StrikeData>
                <Underlying>
                        <Type>Equity</Type>
                        <Name>RIC:.SPX</Name>
                        <Currency>USD</Currency>
                </Underlying>
                <OptionData>
                        <LongShort>Long</LongShort>
                        <OptionType>Call</OptionType>
                        <PayoffType>Asian</PayoffType>
                        <PayoffType2>Arithmetic</PayoffType2>
                        <ExerciseDates>
                                <ExerciseDate>2020-07-15</ExerciseDate>
                         </ExerciseDates>
                         <Premiums> ... </Premiums>
                </OptionData>
                <Settlement>2020-07-20</Settlement>
                <ObservationDates>
                        <Rules>
                                <StartDate>2019-12-27</StartDate>
                                <EndDate>2020-07-06</EndDate>
                                <Tenor>1D</Tenor>
                                <Calendar>US</Calendar>
                                <Convention>F</Convention>
                                <TermConvention>F</TermConvention>
                                <Rule>Forward</Rule>
                        </Rules>
                </ObservationDates>
        </EquityAsianOptionData>
</Trade>
```

In the above example, the holder of the EquityAsianOption has a call option that gives the right but not obligation to pay 310,000 USD (Strike*Quantity) and receive [the averaged equity spot price during the Asian period] USD multiplied by the `Quantity`.

If OptionType would be changed to Put, the holder of the option would have the right to receive 310,000 USD (Strike*Quantity) and pay [the averaged equity spot price during the Asian period] USD multiplied by the `Quantity`.

The payoff is:

$$Payoff = Quantity \cdot MAX(\omega \cdot (A(0,T) - K), 0)$$

where:

- $A(0,T)$: the arithmetic average of underlying euqity spot price over the Asian observation period from start 0 to end T, quoted in `Currency`
- K: equity strike price, quoted in `Currency`
- ω: 1 for a call option (ie receiving averaged equity spot price and paying strike), -1 for a put option

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

- StrikeData: A node containing the strike in `Value` and the currency in which both the underlying and the strike are quoted in `Currency`. Allowable values: See `Currency` in Table 13. The strike may be any positive real number. The currency provided in this node may be quoted as corresponding minor currency to the underlying major currency.
- Quantity: The quantity of the underlying equities. See payoff formula above. Allowable values: all positive real numbers
- Underlying: One (and only one) `Underlying` node where `Type` must be set to *Equity*.
  Allowable values: See 2.3.29. Note that the equity must be quoted in the `Currency` above.
- OptionData: The relevant fields in the `OptionData` node for an EquityAsianOption are the `LongShort` flag, the `OptionType` (*call/put*), the `PayoffType` which must be set to *Asian* or *AverageStrike* to identify a fixed or floating strike asian payoff, and the `ExerciseDates` node where exactly one ExerciseDate date element must be given. `PayoffType2` can be optionally set to *Arithmetic* or *Geometric* and defaults to *Arithmetic* if not given. Furthermore, a *Premiums* node can be added to represent deterministic option premia to be paid by the option holder.
  Allowable values: See 2.3.1 for the general structure of the option data node
- Settlement[Optional]: The settlement date.
  Allowable values: See `Date` in Table 13. Defaults to the ExerciseDate if left blank or omitted.
- ObservationDates: The observation dates for the asian period, given as a rules-based or dates-based schedule, analogous to a `ScheduleData` node but called `ObservationDates`.
  Allowable values: See the definition in 2.3.4

2.2.28 Equity Barrier Option

Payoff

An Equity Barrier Option is a path-dependent option whose existence depends upon an Equity spot rate reaching a pre-set barrier level. Exercise is European.

This product has a continuously monitored single barrier with a Vanilla European Equity Option Underlying. A set number of shares of a single name equity or an

equity index is specified by the “Quantity".

Single Equity Barrier Options can be knock-in or knock-out:

- A knock-in option is a barrier option that only comes into existence/becomes active when the Equity spot rate reaches the barrier level at any point in the option's life. Once a barrier is knocked-in, the option will not cease to exist until the option expires and effectively it becomes a Vanilla Equity Option.
- A knock-out option starts its life active, but ceases to exist/becomes inactive, if the barrier is reached during the life of the option.

When a Single Equity Barrier option expires inactive, the payoff may be zero, or there may be a cash rebate (barrier rebate) paid out as a fraction of the original option premium.

There are four main types of Single Barrier Equity Options:

- Up-and-out: The Equity spot price starts below the barrier level and has to move up for the option to be knocked out.
- Down-and-out: The Equity spot price starts above the barrier level and has to move down for the option to become knocked out.
- Up-and-in: The Equity spot price starts below the barrier level and has to move up for the option to become activated.
- Down-and-in: The Equity spot price starts above the barrier level and has to move down for the option to become activated.

Input

The `EquityBarrierOptionData` node is the trade data container for the *EquityBarrierOption* trade type. The barrier level of an Equity Barrier Option should be quoted in the currency of the underlying Equity spot price. The `EquityBarrierOptionData` node includes one `OptionData` trade component sub-node and one `BarrierData` trade component sub-node plus elements specific to the Equity Barrier Option.

The structure of an example `EquityBarrierOptionData` node for an Equity Barrier Option is shown in Listing 38.

Listing 38: Equity Barrier Option data

```
<EquityBarrierOptionData>
    <OptionData>
        ...
    </OptionData>
    <BarrierData>
        ...
    </BarrierData>
    <StartDate>2025-01-25</StartDate>
    <Calendar>TARGET</Calendar>
    <EQIndex>EQ-RIC:.SPX</EQIndex>
    <Name>RIC:.SPX</Name>
    <StrikeData>
    <StrikePrice>
     <Value>3200.00</Value>
     <Currency>USD</Currency>
    </StrikePrice>
    </StrikeData>
    <Quantity>1000</Quantity>
    <Currency>USD</Currency>
</EquityBarrierOptionData>
```

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

- OptionData: This is a trade component sub-node outlined in section 2.3.1. Note that the Equity Barrier Option type allows for *European* option style only.
- BarrierData: This is a trade component sub-node outlined in section 2.3.31. Level specified in BarrierData should be quoted in the same currency with the underlying Equity spot price. Changing the option from Call to Put or vice versa does not require switching the barrier level.
- StartDate[Optional]: The start date for checking if a barrier has been breached prior to today's date. If omitted or left blank no check is made and it is assumed no barrier has been breached in the past. Has no impact if set to today's date or a date in the future.

  Allowable values: See `Date` in Table 13.
- Calendar[Optional]: The calendar associated with the Equity Index. Required if StartDate is set to a date prior to today's date, otherwise optional.

  Allowable values: See Table 17 Calendar.
- EQIndex[Optional]: A reference to an Equity Index source to check if the barrier has been breached. Required if StartDate is set to a date prior to today's date, otherwise optional and can be omitted but not left blank.

  Allowable values: The format of the Equity Index is"EQ-RIC:Code".
- Underlying: This node may be used as an alternative to the `Name` node to specify the underlying equity. This in turn defines the equity curve used for pricing. The `Underlying` node is described in further detail in Section 2.3.29.

- StrikeData: A node containing the strike in `Value` and the currency in which both the underlying and the strike are quoted in `Currency`. I.e. compo options with strike currency not equal to underlying equity currency are not supported for this trade type.

  Allowable values: Only supports `StrikePrice` as described in Section 2.3.30.

- Quantity: The number of units of the underlying covered by the transaction.

  Allowable values: Any positive real number.

- Currency: The payment currency of the trade.

  Allowable values: See `Currency` and `Minor Currencies` in Table 13. This should be equal to the underlying equity except for the major / minor distinction. I.e. quanto payoffs that are usually identified by setting the payment currency to a different currency than the underlying equity currency, are not allowed for this trade type.

2.2.29 Equity Digital Option

Payoff

An Equity Digital Option is an option whose payout is either zero or a fixed predetermined amount (Cash-or-Nothing). Payout depends on whether the underlying Equity spot rate expires in-the-money at the expiration date.

- Equity Digital Options have European exercise with payout at expiry.
- The buyer of a Equity Digital Option pays a premium to the seller.

Input

The `EquityDigitalOptionData` node is the trade data container for the *EquityDigitalOption* trade type. The `EquityDigitalOptionData` node includes one `OptionData` trade component sub-node plus elements specific to the Equity Digital Option. The structure of an example `EquityDigitalOptionData` node for an Equity Digital Option is shown in Listing 39.

Listing 39: Equity Digital Option data

```
<EquityDigitalOptionData>
    <OptionData>
        <LongShort>Long</LongShort>
        <OptionType>Call</OptionType>
        <Style>European</Style>
        <ExerciseDates>
            <ExerciseDate>2027-02-26</ExerciseDate>
        </ExerciseDates>
        ...
    </OptionData>
    <Strike>3300</Strike>
    <PayoffCurrency>USD</PayoffCurrency>
    <PayoffAmount>1000</PayoffAmount>
    <Name>RIC:.SPX</Name>
    <Quantity>1000</Quantity>
</EquityDigitalOptionData>
```

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

- OptionData: This is a trade component sub-node outlined in section 2.3.1. The relevant fields in the `OptionData` node for an EquityDigitalOption are:
  - `LongShort` The allowable values are *Long* or *Short*.
  - `OptionType` The allowable values are *Call* or *Put*. *Call* means that the option is in the money when the underlying equity price is above the strike, and *Put* means that the option is in the money when the underlying equity price is below the strike.
  - `Style` The allowable value is *European*. Note that the Equity Digital Option type allows for *European* option style only.
  - An `ExerciseDates` node where exactly one ExerciseDate date element must be given.
  - Premiums [Optional]: Option premium amounts paid by the option buyer to the option seller.

    Allowable values: See section 2.3.2
- Strike: The option strike price per one unit of the underlying, expressed in the currency of the underlying equity .

  Allowable values: Any positive real number.
- PayoffCurrency: The payoff currency of the Equity Digital Option is the currency of the payoff amount. Must be consistent with the currency of the underlying Equity spot price.

  Allowable values: See Table 15 `Currency`.
- PayoffAmount: The fixed payoff amount per unit of underlying expressed in payoff currency. It is cash-or-nothing payoff that depends on the option being in or out of the money.

Allowable values: Any positive real number.

- Underlying: This node may be used as an alternative to the `Name` node to specify the underlying equity. This in turn defines the equity curve used for pricing. The `Underlying` node is described in further detail in Section 2.3.29.
- Quantity: The number of units of the underlying covered by the transaction.

  Allowable values: Any positive real number.

2.2.30 Equity Double Barrier Option

The `EquityDoubleBarrierOptionData` node is the trade data container for the *EquityDoubleBarrierOption* trade type.

An Equity Double Barrier Option is a path-dependent option whose existence depends upon an Equity spot rate reaching one of the two pre-set barrier levels. Exercise is European, and barriers are American (continuously monitored).

Equity Double Barrier options can be knock-in or knock-out:

- A knock-in option is a barrier option that only comes into existence/becomes active when the Equity spot rate reaches the one of the barrier level at any point in the option's life. Once a barrier is knocked-in, the option will not cease to exist until the option expires and effectively it becomes a Vanilla Equity Option.
- A knock-out option starts its life active, but ceases to exist/becomes inactive, if the one of the barriers is reached during the life of the option.

The barrier levels of an Equity Double Barrier Option should be quoted in the currency of the underlying Equity spot price. The `EquityDoubleBarrierOptionData` node includes one `OptionData` trade component sub-node and one `BarrierData` trade component sub-node plus elements specific to the Equity Double Barrier Option. The structure of an example `EquityDoubleBarrierOptionData` node for a Equity Double Barrier Option is shown in Listing 40.

Listing 40: Equity Double Barrier Option data

```
<EquityDoubleBarrierOptionData>
    <OptionData>
        <LongShort>Long</LongShort>
        <OptionType>Call</OptionType>
        <Style>European</Style>
        <Settlement>Cash</Settlement>
        <ExerciseDates>
            <ExerciseDate>2021-01-29</ExerciseDate>
        </ExerciseDates>
    </OptionData>
    <BarrierData>
        <Type>KnockOut</Type>
        <Levels>
            <Level>3000.00</Level>
            <Level>3500.00</Level>
        </Levels>
    </BarrierData>
    <Name>RIC:.SPX</Name>
    <Currency>USD</Currency>
            <StrikeData>
                    <StrikePrice>
                            <Value>3200.00</Value>
                            <Currency>USD</Currency>
                    </StrikePrice>
            </StrikeData>
    <Quantity>1000</Quantity>
    <StartDate>2019-12-27</StartDate>
    <Calendar>US-NYSE</Calendar>
    <EQIndex>EQ-RIC:.SPX</EQIndex>
</EquityDoubleBarrierOptionData>
```

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

- OptionData: This is a trade component sub-node outlined in section 2.3.1. The relevant fields in the `OptionData` node for an EquityDoubleBarrierOption are:
  - `LongShort` The allowable values are *Long* or *Short*.
  - `OptionType` The allowable values are *Call* or *Put*.
  - `Style` The Equity Double Barrier Option type allows for *European* option exercise style only.
  - `Settlement` The allowable values are *Cash* or *Physical*.
  - An `ExerciseDates` node where exactly one `ExerciseDate` date element must be given.
  - Optional `PremiumAmount`, `PremiumCurrency`, and `PremiumPayDate` fields to specify the Equity Double Barrier Option premium.
- BarrierData: This is a trade component sub-node outlined in section 2.3.31. Two levels in ascending order should be defined in `Levels`. `Type` should be *KnockOut* or *KnockIn*.

Allowable values: See Table 17 Calendar.

- Underlying: This node may be used as an alternative to the `Name` node to specify the underlying equity. This in turn defines the equity curve used for pricing. The `Underlying` node is described in further detail in Section 2.3.29.
- Currency: The currency of the equity option.

  Allowable values: See Table 15 `Currency`.
- StrikeData: A node containing the strike in `Value` and the currency in which both the underlying and the strike are quoted in `Currency`. Allowable values: Only supports `StrikePrice` as described in Section 2.3.30.
- Quantity: The number of units of the underlying covered by the transaction.

  Allowable values: Any positive real number.
- StartDate [Optional]: The start date for checking if a barrier has been breached prior to today's date. If omitted or left blank no check is made and it is assumed no barrier has been breached in the past. Has no impact if set to today's date or a date in the future.

  Allowable values: See `Date` in Table 13.
- Calendar [Optional]: The calendar associated with the Equity Index. Required if StartDate is set to a date prior to today's date, otherwise optional.
- EQIndex [Optional]: A reference to an Equity Index source to check if the barrier has been breached. Required if StartDate is set to a date prior to today's date, otherwise optional and can be omitted but not left blank.

  Allowable values: The format of the Equity Index is"EQ-RIC:Code".

2.2.31 Equity Double Touch Option

Payoff

This product has two continuously monitored barriers with a Cash-or-Nothing digital underlying.

An Equity Double Touch option pays either a fixed predetermined payoff amount or zero (Cash-or-Nothing) depending upon an Equity spot rate reaching one of the pre-set barrier levels.

- Can be of the same two main types as a regular Barrier Equity Option: Knock-In, Knock-Out.
- A Knock-In or Double One-Touch option has a fixed payout if one of the barriers is breached, and no payout otherwise.
- A Knock-Out or Double No-Touch option has no payout if one of the barriers is breached, and fixed payout otherwise.
- The buyer of a Equity Double Touch Option pays a premium to the seller.

Note that the current implementation supports Equity Double Touch options with payout at expiry only.

Input

The `EquityDoubleTouchOptionData` node is the trade data container for the *EquityDoubleTouchOption* trade type. The `EquityDoubleTouchOptionData` node includes one `OptionData` trade component sub-node and one `BarrierData` trade component sub-node plus elements specific to the Equity Double Touch Option.

The structure of an example `EquityDoubleTouchOptionData` node for an Equity Double Touch Option is shown in Listing 41.

Listing 41: Equity Double Touch Option data

```
<EquityDoubleTouchOptionData>
    <OptionData>
        <LongShort>Long</LongShort>
        <PayOffAtExpiry>true</PayOffAtExpiry>
        <ExerciseDates>
         <ExerciseDate>2021-12-14</ExerciseDate>
        </ExerciseDates>
        ...
    </OptionData>
    <BarrierData>
        ...
        <Type>KnockIn</Type> <!-- KnockOut or KnockIn -->
        <Levels>
            <Level>3000</Level>
            <Level>4500</Level>
        </Levels>
        ...
    </BarrierData>
    <PayoffCurrency>USD</PayoffCurrency>
    <PayoffAmount>1000000</PayoffAmount>
    <Name>RIC:.SPX</Name>
    <StartDate>2021-03-01</StartDate>
    <Calendar>USD</Calendar>
    <EQIndex>EQ-RIC:.SPX</EQIndex>
</EquityDoubleTouchOptionData>
```

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

- OptionData: This is a trade component sub-node outlined in section 2.3.1. The relevant fields in the `OptionData` node for an EquityDoubleTouchOption are as below. Note that the `OptionType` can be omitted.
  - `LongShort` The allowable values are *Long* or *Short*.
  - `PayOffAtExpiry` [Optional] *true* for payoff at expiry and *false* for payoff at hit. Currently, for both *KnockOut* and *KnockIn* barriers, only payoff at expiry (i.e. *true*) is supported. Defaults to *true* if left blank or omitted.
  - An `ExerciseDates` node where exactly one ExerciseDate date element must be given.
  - Premiums [Optional]: Option premium amounts paid by the option buyer to the option seller.

Allowable values: See section 2.3.2

- BarrierData: This is a trade component sub-node outlined in section 2.3.31. Two levels in ascending order should be defined in *Levels*. *Type* should be *KnockOut* or *KnockIn*. Levels specified in BarrierData should be quoted in the same currency as the underlying Equity spot prices.
- PayoffCurrency: The payoff currency of the Equity Double Touch Option is the currency of the payoff amount. Must be consistent with the currency of the underlying Equity spot prices.

  Allowable values: See Table 15 `Currency`.
- PayoffAmount: The fixed payoff amount expressed in payoff currency. It is cash-or-nothing payoff that depends on the option being in or out of the money, and whether the barrier has been touched.

  Allowable values: Any positive real number.
- Underlying: This node may be used as an alternative to the `Name` node to specify the underlying equity. This in turn defines the equity curve used for pricing. The `Underlying` node is described in further detail in Section 2.3.29.
- StartDate[Optional]: The start date for checking if a barrier has been breached prior to today's date. If omitted or left blank no check is made and it is assumed no barrier has been breached in the past. Has no impact if set to today's date or a date in the future.

  Allowable values: See `Date` in Table 13.
- Calendar[Optional]: The calendar associated with the Equity Index. Required if StartDate is set to a date prior to today's date, otherwise optional.

  Allowable values: See Table 17 Calendar.
- EQIndex[Optional]: A reference to an Equity Index source to check if the barrier has been breached. Required if StartDate is set to a date prior to today's date, otherwise optional and can be omitted but not left blank.

  Allowable values: The format of the Equity Index is"EQ-RICCode".

2.2.32 Equity European Barrier Option

Payoff

An Equity European Barrier option gives the buyer the right, but not the obligation, to buy a set number of shares of a single name equity or an equity index, at a predetermined strike price, at one predetermined time in the future. This right may be withdrawn depending upon an Equity spot price or index reaching a predetermined barrier level at the predetermined time, the underlying is monitored only at expiry with a single barrier.

For this right the buyer pays a premium to the seller. Settlement can be either cash or physical delivery.

Single Equity European Barrier options can be knock-in or knock-out:

- A knock-in option is a barrier option that only comes into existence/becomes active when the Equity spot rate reaches the barrier level at expiry.
- A knock-out option starts its life active, but ceases to exist/becomes inactive, if the barrier is reached at expiry.

When a Single Equity European Barrier option expires inactive, the payoff may be zero, or there may be a cash rebate (barrier rebate) paid out as a fraction of the original option premium.

There are four main types of Single European Barrier Equity Options:

- Up-and-out: The Equity spot price starts below the barrier level and has to move up for the option to be knocked out.
- Down-and-out: The Equity spot price starts above the barrier level and has to move down for the option to become knocked out.
- Up-and-in: The Equity spot price starts below the barrier level and has to move up for the option to become activated.
- Down-and-in: The Equity spot price starts above the barrier level and has to move down for the option to become activated.

Input

The `EquityEuropeanBarrierOptionData` node is the trade data container for the *EquityEuropeanBarrierOption* trade type. The barrier level of an Equity European Barrier Option is quoted in the currency of the underlying Equity spot price. The `EquityEuropeanBarrierOptionData` node includes one `OptionData` trade component sub-node and one `BarrierData` trade component sub-node plus elements specific to the Equity European Barrier Option.

The structure of an example `EquityEuropeanBarrierOptionData` node for an Equity European Barrier Option is shown in Listing 42.

Listing 42: Equity European Barrier Option data

```
<EquityEuropeanBarrierOptionData>
    <OptionData>
        ...
    </OptionData>
    <BarrierData>
        ...
    </BarrierData>
    <Name>RIC:.SPX</Name>
    <StrikeData>
        <StrikePrice>
                                <Value>3200.00</Value>
                                <Currency>USD</Currency>
                        </StrikePrice>
    </StrikeData>
    <Quantity>1000</Quantity>>
</EquityEuropeanBarrierOptionData>
```

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

- OptionData: This is a trade component sub-node outlined in section 2.3.1. Note that the Equity European Barrier Option type allows for *European* option style only.
- BarrierData: This is a trade component sub-node outlined in section 2.3.31. Level specified in BarrierData should be quoted in the same currency with the underlying Equity spot price. Changing the option from Call to Put or vice versa does not require switching the barrier level.
- Underlying: This node may be used as an alternative to the `Name` node to specify the underlying equity. This in turn defines the equity curve used for pricing. The `Underlying` node is described in further detail in Section 2.3.29.
- StrikeData: A node containing the strike in `Value` and the currency in which both the underlying and the strike are quoted in `Currency`. Allowable values: Only supports `StrikePrice` as described in Section 2.3.30.
- Quantity: The number of units of the underlying covered by the transaction.

  Allowable values: Any positive real number.

2.2.33 Equity Touch Option

Payoff

This product has a continuously monitored single barrier with a Cash-or-Nothing digital underlying.

At expiry an Equity Touch Option pays either a fixed predetermined payoff amount or zero (Cash-or-Nothing) depending upon an Equity spot rate reaching a pre-set barrier level.

- Can be of the same four main types as a regular Barrier Equity Option: Up-and-out, Down-and-out, Up-and-in, Down-and-in.
- A Knock-In or One-Touch option has a fixed payout if the barrier is breached, and no payout otherwise. The payout can be at expiry or at hit.
- A Knock-Out or No-Touch option has no payout if the barrier is breached, and fixed payout at expiry otherwise.
- The buyer of a Equity Touch Option pays a premium to the seller.

Input

The `EquityTouchOptionData` node is the trade data container for the *EquityTouchOption* trade type. The `EquityTouchOptionData` node includes one `OptionData` trade component sub-node and one `BarrierData` trade component sub-node plus elements specific to the Equity Touch Option.

The structure of an example `EquityTouchOptionData` node for an Equity Touch Option is shown in Listing 43.

Listing 43: Equity Touch Option data

```
<EquityTouchOptionData>
    <OptionData>
        <LongShort>Long</LongShort>
        <PayOffAtExpiry>true</PayOffAtExpiry>
        <Settlement>Cash</Settlement>
        <ExerciseDates>
         <ExerciseDate>2022-03-01</ExerciseDate>
        </ExerciseDates>
        ...
    </OptionData>
    <BarrierData>
       <Type>UpAndIn</Type>
       <Levels>
           <Level>3300</Level>
       </Levels>
    </BarrierData>
    <PayoffCurrency>USD</PayoffCurrency>
    <PayoffAmount>1000000</PayoffAmount>
    <Name>RIC:.SPX</Name>
    <StartDate>2019-12-27</StartDate>
    <Calendar>US-NYSE</Calendar>
    <EQIndex>EQ-RIC:.SPX</EQIndex>>
</EquityTouchOptionData>
```

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

- OptionData: This is a trade component sub-node outlined in section 2.3.1. The `OptionType` sub-node is not required and is inferred from the `BarrierData` type (i.e. *Call* for an Up barrier, and *Put* for a Down barrier). The relevant fields in the `OptionData` node for an EquityTouchOption are:
  - `LongShort` The allowable values are *Long* or *Short*.
  - `PayOffAtExpiry` [Optional] *true* for payoff at expiry and *false* for payoff at hit. For *UpAndOut* and *DownAndOut* barriers, only payoff at expiry (i.e. *true*) is supported. Defaults to *true* if left blank or omitted.
  - `Settlement` The allowable values are *Cash* or *Physical*.
  - An `ExerciseDates` node where exactly one ExerciseDate date element must be given.
  - Premiums [Optional]: Option premium amounts paid by the option buyer to the option seller.

    Allowable values: See section 2.3.2
- BarrierData: This is a trade component sub-node outlined in section 2.3.31. Level specified in BarrierData should be quoted in the same currency as the underlying Equity spot price.
- PayoffCurrency: The payoff currency of the Equity Touch Option is the currency of the payoff amount. Must be consistent with the currency of the underlying

Equity spot price.

Allowable values: See `Currency` in Table 13.

- PayoffAmount: The fixed payoff amount expressed in payoff currency. It is cash-or-nothing payoff that depends on the option being in or out of the money, and whether the barrier has been touched.

  Allowable values: Any positive real number.

- Underlying: This node may be used as an alternative to the `Name` node to specify the underlying equity. This in turn defines the equity curve used for pricing. The `Underlying` node is described in further detail in Section 2.3.29.

- StartDate[Optional]: The start date for checking if a barrier has been breached prior to today's date. If omitted or left blank no check is made and it is assumed no barrier has been breached in the past. Has no impact if set to today's date or a date in the future.

  Allowable values: See `Date` in Table 13.

- Calendar[Optional]: The calendar associated with the Equity Index. Required if StartDate is set to a date prior to today's date, otherwise optional.

  Allowable values: See Table 17 Calendar.

- EQIndex[Optional]: A reference to an Equity Index source to check if the barrier has been breached. Required if StartDate is set to a date prior to today's date, otherwise optional and can be omitted but not left blank.

  Allowable values: The format of the Equity Index is"EQ-RICCode".

2.2.34 Equity Variance Swap

Payoff

An Equity Variance Swap has a payoff that depends on the volatility of an underlying equity instrument. Underlying individual stocks and indices are supported. The swap counterparties agree to exchange a pre-agreed variance level (the strike) for the actual amount of variance realized over an observation period.

The strike is typically set at ATM so the swap initially has zero value. If the subsequent realized volatility is above the strike level, the buyer of a variance swap who is long volatility will have a positive NPV, and the seller who is short volatility, will have a negative NPV.

Payoff:

$$N \cdot \left(\text{RealisedVol}^2 - K^2\right)$$

where

- N: Notional, also called Variance Notional, determined as Vega Notional/$(2K)$.
- Vega Notional: Notional in terms of volatility units.

- RealisedVol:

$$\sqrt{252 \cdot \sum_{t=1}^{N} \frac{1}{\text{TradingDays}} \left(\ln \frac{P_t}{P_{t-1}}\right)^2} \cdot 100$$

- TradingDays: the number of days which are expected to be scheduled trading days in the observation period
- P_0: the official closing of the underlying at the observation start date
- P_t: the official closing of the underlying at any observation date t, or at observation end date
- K: the strike volatility

Input

The `EqutiyVarianceSwapData` node is the trade data container for the *EquityVarianceSwap* trade type. Only vanilla variance swaps are supported. The structure of an example `EqutiyVarianceSwapData` node for an equity variance swap is shown in Listing 44.

Listing 44: Variance Swap data

```
<EquityVarianceSwapData>
    <StartDate>2016-01-29</StartDate>
    <EndDate>2016-05-05</EndDate>
    <Currency>USD</Currency>
    <Underlying>
      <Type>Equity</Type>
      <Name>.SPX</Name>
      <IdentifierType>RIC</IdentifierType>
    </Underlying>
    <LongShort>Long</LongShort>
    <Strike>0.20</Strike>
    <Notional>50000</Notional>
    <Calendar>US</Calendar>
    <MomentType>Variance</MomentType>
    <AddPastDividends>true</AddPastDividends>
</EqutiyVarianceSwapData>
```

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

- StartDate: The variance swap start date.
  Allowable values: See `Date` in Table 13.
- EndDate: The variance swap end date.
  Allowable values: See `Date` in Table 13.
- Currency: The bought currency of the variance swap.
  Allowable values: See `Currency` in Table 13.
- Name: The identifier of the underlying equity or equity index.
  Allowable values: See `Name` for equity trades in Table 23.

- Underlying: This node may be used as an alternative to the `Name` node to specify the underlying equity. This in turn defines the equity curve used for pricing. The `Underlying` node is described in further detail in Section 2.3.29.
- LongShort: Defines whether the trade is long in the equity variance. For the avoidance of doubt, a long variance swap has positive value if the realised variance exceeds the variance strike.
  Allowable values: *Long, Short*
- Strike: The volatility strike K_{vol} of the variance swap quoted absolutely (i.e. not as a percent). If the swap was struck in terms of variance, the square root of that variance should be used here.
  Allowable values: Any positive real number.
- Notional: The vega notional of the variance swap. This is the notional in terms of volatility units (like the strike). If the swap was struck in terms of a variance notional N_{var}, the corresponding vega notional is given by $N_{vol} = N_{var} * 2 * 100 * K_{vol}$ (where K_{vol} is in absolute terms).
  Allowable values: Any non-negative real number.
- Calendar: The calendar determining the observation/fixing dates according to which variance is accrued is the combination of the calendar(s) given here plus the calendar associated with the equity in the equity curve configuration. If no such calendar is given in the equity curve configuration the standard calendar for the equity currency (also defined in the curve config) is used instead.
  Allowable values: See Table 17.
- MomentType[Optional]: A flag to distinguish if the swap is struck in terms of volatility or variance. The MomentType should be set to *Volatility* or *Variance* depending on the payoff. Note that MomentType does not necessarily need to be equivalent to the way the Strike is quoted which is always as a Volatility.
  Allowable values: *Volatility* or *Variance*. Defaults to *Variance* if left blank or omitted.
- AddPastDividends[Optional]: A flag to distinguish if past dividend payments should be added to the fixings when calculating accrued variance.
  Allowable values: *true* or *false*. Defaults to *false* if left blank or omitted.

2.2.35 Equity Cliquet Option

Payoff

A cliquet option is an exotic option consisting of a series of consecutive forward start options, with each option being struck at-the-money when it becomes active. The Cliquet Option's payoff is:

$$N \cdot \min\left(cap_g, \max\left(floor_g, \sum_{i=1}^{n} \delta \cdot \min\left(cap_l, \max\left(floor_l, S_{t_i}/S_{t_{i-1}} - M\right)\right)\right)\right)$$

where

- S_{t_i}: Price of the underlying at time t_t.

- cap_g: Global Cap.
- $floor_g$: Global Floor.
- cap_l: Local Cap.
- $floor_l$: Local Floor.
- δ: 1 for Call, -1 for Put option.
- M: Moneyness.
- n: Number of valuation dates.
- N: Notional

Input

The `EquityCliquetOptionData` node is the trade data container for the *EquityCliquetOption* trade type. A cliquet option consists of a series of consecutive forward starting equity options, with each option being struck at a given moneyness (commonly at-the-money) when it becomes active.

The structure of an example `EquityCliquetOptionData` node for an equity cliquet option is shown in Listing 45.

Listing 45: Cliquet Option data

```
<EquityCliquetOptionData>
    <Underlying>
      <Type>Equity</Type>
      <Name>.SPX</Name>
      <IdentifierType>RIC</IdentifierType>
    </Underlying>
    <Currency>USD</Currency>
    <Notional>1000000.0</Notional>
    <LongShort>Short</LongShort>
    <OptionType>Call</OptionType>
    <Moneyness>1.0</Moneyness>
    <LocalCap>0.07</LocalCap>
    <LocalFloor>-0.06</LocalFloor>
    <GlobalCap>0.07</GlobalCap>
    <GlobalFloor>-0.07</GlobalFloor>
    <ScheduleData>
        <Dates>
            <Dates>
                <Date>20171231</Date>
                <Date>20181231</Date>
                <Date>20191231</Date>
                <Date>20201231</Date>
                <Date>20211231</Date>
                <Date>20221231</Date>
            </Dates>
            <Calendar>USD</Calendar>
            <Convention>F</Convention>
        </Dates>
    </ScheduleData>
    <SettlementDays>5</SettlementDays>
    <Premium>0.027</Premium>
    <PremiumPaymentDate>31-12-2017</PremiumPaymentDate>
    <PremiumCurrency>USD</PremiumCurrency>
</EquityCliquetOptionData>
```

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

- Name: The identifier of the underlying equity or equity index.
  Allowable values: See `Name` for equity trades in Table 23.
- Underlying: This node may be used as an alternative to the `Name` node to specify the underlying equity. This in turn defines the equity curve used for pricing. The `Underlying` node is described in further detail in Section 2.3.29.
- Currency: The currency of the notional, and thus of the option.
  Allowable values: See `Currency` in Table 13. The Currency must be the same as the currency of the underlying equity.
- Notional: The notional of the cliquet option.
  Allowable values: Any positive real number.
- LongShort: Defines whether the trade is long or short the option.
  Allowable values: *Long, Short*

- OptionType: The type of the option.
  Allowable values: *Call, Put*
- Moneyness: Adjustment of option return. The moneyness M each forward starting option is being struck at.
  Allowable values: Any real number. Expressed in decimal form where 1.0 is at-the-money, 1.1 is 110% of the at-the-money strike, 0.9 is 90% of the at-the-money strike, etc.
- LocalCap[Optional]: The local cap, cap_l, in each of the option return.
  Allowable values: Any real number. If omitted, no local cap is applied. Can't be left blank.
- LocalFloor[Optional]: The local floor, $floor_l$, in each of the option return.
  Allowable values: Any real number. If omitted, no local floor is applied. Can't be left blank.
- GlobalCap[Optional]: The global cap, cap_g, for the option return.
  Allowable values: Any real number. If omitted, no global cap is applied. Can't be left blank.
- GlobalFloor[Optional]: The global floor,$floor_g$, for the option return.
  Allowable values: Any real number. If omitted, no global floor is applied. Can't be left blank.
- ScheduleData: A schedule of dates that define the valuation dates of the consecutive forward starting options forming the Equity Cliquet Option. The first date in the schedule is the start date of the first consecutive option, the second date in the schedule is the end/valuation date of the first consecutive option, and also the start date of the second consecutive option, etc. The last date is the final valuation date, with payoff of the whole Cliquet option at this date plus `SettlementDays`.
  Allowable values: A node on the same form as `ScheduleData`, (see 2.3.4).
- SettlementDays[Optional]: Number of days from the last valuation date to the payoff being paid or received. The payoff date is determined with regards to calendar and term date convention of the schedule's calendar.
  Allowable values: Any positive integer. Defaults to zero if left blank or omitted.
- Premium[Optional]: The premium paid for the option.
  Allowable values: Any real number. Expressed in decimal form relative to notional.
- PremiumPaymentDate[Optional]: The date the premium is the paid.
  Allowable values: See `Date` in Table 13. Note that if a Premium is specified, a PremiumPaymentDate must also be specified.
- PremiumCurrency[Optional]: The currency the premium is to paid in.
  Allowable values: See `Currency` in Table 13. Defaults to the currency of the notional.

2.2.36 Equity Position

An equity position represents a position in a single equity - using a single `Underlying` node, or in a weighted basket of underlying equities - using multiple `Underlying` nodes.

An Equity Position can be used both as a stand alone trade type (TradeType: *EquityPosition*) or as a trade component (`EquityPositionData`) used within the *TotalReturnSwap* (Generic TRS) trade type, to set up for example Equity Basket trades.

It is set up using an `EquityPositionData` block as shown in listing 46. The meanings and allowable values of the elements in the block are as follows:

- Quantity: The number of shares or units of the weighted basket held.
  Allowable values: Any positive real number
- Underlying: One or more underlying descriptions. If a basket of equities is defined, the `Weight` field should be populated for each underlyings. The weighted basket price is then given by

$$\text{Basket-Price} = \text{Quantity} \times \sum_i \text{Weight}_i \times S_i \times \text{FX}_i$$

  where

  - S_i is the price of the ith share in the basket
  - FX_i is the FX Spot converting from the ith equity currency to the first equity currency which is by definition the currency in which the npv of the basket is expressed.

  Allowable values: See 2.3.29 for the definition of an underlying. Only equity underlyings are allowed.

Listing 46: Equity position data

```
<Trade id="EquityPosition">
  <TradeType>EquityPosition</TradeType>
  <Envelope>...</Envelope>
  <EquityPositionData>
    <Quantity>1000</Quantity>
      <Underlying>
        <Type>Equity</Type>
        <Name>BE0003565737</Name>
        <Weight>0.5</Weight>
        <IdentifierType>ISIN</IdentifierType>
        <Currency>EUR</Currency>
        <Exchange>XFRA</Exchange>
      </Underlying>
      <Underlying>
        <Type>Equity</Type>
        <Name>GB00BH4HKS39</Name>
        <Weight>0.5</Weight>
        <IdentifierType>ISIN</IdentifierType>
        <Currency>GBP</Currency>
        <Exchange>XLON</Exchange>
      </Underlying>
  </EquityPositionData>
</Trade>
```

2.2.37 Equity Option Position

An equity option position represents a position in a single equity option - using a single `Underlying` node, or in a weighted basket of underlying equity options - using multiple `Underlying` nodes.

An Equity Option Position can be used both as a stand alone trade type (TradeType: *EquityOptionPosition*) or as a trade component (`EquityOptionPositionData`) used within the *TotalReturnSwap* (Generic TRS) trade type, to set up for example Equity Option Basket trades.

It is set up using an `EquityOptionPositionData` block as shown in listing 47. The meanings and allowable values of the elements in the block are as follows:

- Quantity: The number of options written on one underlying share resp. the number of units of the option basket held.
  Allowable values: Any positive real number
- Underlying: One or more underlying descriptions, each comprising an `Underlying` block, an `Optiondata` block and a `Strike` element, in that order:
  - Underlying: an underlying description, see 2.3.29, only equity underlying are allowed
  - OptionData: the option description, see 2.3.1, the relevant / allowed data is
    - LongShort: the type of the position,*long* and *Short* positions are allowed. Note that negative weights are allowed. A *long* position with a negative weight results in a *short* position, and a *short* position with a negative weight results in a *long* position.

        * OptionType: *Call* or *Put*
        * Style: *European* or *American*
        * Settlement: *Cash* or *Physical*
        * ExerciseDates: exactly one exercise must be given representing the European exercise date or the last American exercise date
    - Strike: the strike of the option. Allowable values are non-negative real numbers.

If a basket of equities is defined, the `Weight` field should be populated for each underlying. The weighted basket price is then given by

$$\text{Basket-Price} = \text{Quantity} \times \sum_i \text{Weight}_i \times p_i \times \text{FX}_i$$

where

- p_i is the price of the ith option in the basket, written on one underlying share
- FX_i is the FX Spot converting from the ith equity currency to the first equity currency which is by definition the currency in which the npv of the basket is expressed.

Listing 47: Equity Option position data

```
<Trade id="EquityOptionPositionTrade">
  <TradeType>EquityOptionPosition</TradeType>
  <EquityOptionPositionData>
    <!-- basket price = quantity x sum_i ( weight_i x equityOptionPrice_i x fx_i ) -->
    <Quantity>1000</Quantity>
    <!-- option #1 -->
    <Underlying>
      <Underlying>
        <Type>Equity</Type>
        <Name>.SPX</Name>
        <Weight>0.5</Weight>
        <IdentifierType>RIC</IdentifierType>
      </Underlying>
      <OptionData>
        <LongShort>Long</LongShort>
        <OptionType>Call</OptionType>
        <Style>European</Style>
        <Settlement>Cash</Settlement>
        <ExerciseDates>
          <ExerciseDate>2021-01-29</ExerciseDate>
        </ExerciseDates>
      </OptionData>
      <Strike>3300</Strike>
    </Underlying>
    <!-- option #2 -->
    <Underlying>
      <Underlying>
        <Type>Equity</Type>
        <Name>.SPX</Name>
        <Weight>0.5</Weight>
        <IdentifierType>RIC</IdentifierType>
      </Underlying>
      <OptionData>
        <LongShort>Long</LongShort>
        <OptionType>Call</OptionType>
        <Style>European</Style>
        <Settlement>Cash</Settlement>
        <ExerciseDates>
          <ExerciseDate>2021-01-29</ExerciseDate>
        </ExerciseDates>
      </OptionData>
      <Strike>3400</Strike>
    </Underlying>
    <!-- option #3 -->
    <!-- ... -->
  </EquityOptionPositionData>
</Trade>
```

2.2.38 CPI Swap

Payoff

A CPI swap is an inflation swap where one of the legs has a floating rate with coupon payments linked to a supported inflation index.

Coupons on the inflation leg are calculated starting by the contractual real coupon

rate and adjusting it to a nominal rate using the change from the relevant inflation index fixing before issue date to the index fixing at coupon reset date, taking into account observation lag, and if necessary, interpolation between inflation index fixings.

Note that the amount to be disbursed on the maturity date of the inflation leg (excluding the last coupon) is calculated by multiplying the initial notional (face value) value by the increase in the relevant inflation index over the life of the swap. A CPI Swap coupon payment at time i:

$$N\, r\, \frac{I(T_i)}{I(T_0)}\, \delta(T_{i-1}, T_i)$$

where:

- N: notional
- r: the contractual real rate
- $I(T_i)$: the relevant CPI fixing for time T_i
- $I(T_0)$: the relevant CPI fixing before issue date
- $\delta(T_{i-1}, T_i)$: the day count fraction for the accrual period up to time T_i

The flow at maturity excluding the last coupons can be either no flow on both legs, or the notional on the non-inflation leg and the following on the inflation leg:

$$N\, \frac{I(T)}{I(T_0)}$$

where $I(T)$ is the relevant CPI fixing for the maturity date.

CPI linked coupons and maturity cashflows can be capped/floored.

Input

A CPI inflation swap can be set up using the *InflationSwap* trade type, with one leg of type `CPI`. and the other leg(s) can be of any leg type. Listing 48 shows an example. The CPI leg contains an additional `CPILegData` block. See 2.3.17 for details on the CPI leg specification.

Note that Cross Currency Inflation Swaps are supported, as the currencies on the legs of an *InflationSwap* do not need to be the same.

Listing 48: CPI Swap Data (using InflationSwap *trade type)*

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

Alternatively, a CPI swap can be set up as a swap with trade type *Swap*, with one leg of type `CPI`, see listing 49.

Listing 49: CPI Swap Data (using Swap *trade type)*

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

2.2.39 Year on Year Inflation Swap

Payoff

A YoYIIS is a swap contract where one leg has annual inflation linked coupon payments that are exchanged for fixed payments on the other leg.
Inflation-linked payment:

$$N \cdot \left(\frac{I(T_i)}{I(T_{i-1})} - 1 \right) \cdot \delta(T_{i-1}, T_i)$$

where T_{i-1} and T_i are spaced one year apart.

Year-on-year inflation linked coupons can be capped/floored.

Input

A Year on Year inflation swap can be set up with trade type *Swap*, with one leg of type `YY`. Listing 50 shows an example. The YY leg contains an additional `YYLegData` block. See 2.3.18 for details on the YY leg specification.

Listing 50: Year on Year Swap Data (using Swap *trade type)*

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

Alternatively, a Year on Year inflation swap can be set up using the *InflationSwap* trade type, see Listing 51. The structure of the `InflationSwapData` container is the same as for `SwapData` above.

Listing 51: Year on Year Swap Data (using InflationSwap *trade type)*

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

2.2.40 Bond

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

A Bond can be set up in a short version referencing an underlying bond static, or in a long version where the underlying bond details are specified explicitly, including a full LegData block. The short version is shown in listing 52. The details of the bond are

read from the reference data in this case using the SecurityId as a key. The bond trade is fully specified by

- SecurityId: The id identifying the bond.

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

- BondNotional: The notional of the position in the reference bond, expressed in the currency of the bond.

  Allowable Values: Any non-negative real number

- CreditRisk [Optional] Boolean flag indicating whether to show Credit Risk on the Bond product. If set to *false*, the product class will not be set to *Credit*, and there will be no credit sensitivities. However, if the underlying bond reference is set up without a CreditCurveId - typically for some highly rated government bonds - the CreditRisk flag will have no impact on the product class and no credit sensitivities will be shown even if CreditRisk is set to *true*.

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

in this case.

Listing 52: Bond Data

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

For the long version, the bond details are inlined in the trade as shown in listing 53. The bond specific elements are

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

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

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

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

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

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

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

  Allowable values: For currencies with available ibor indices:
  An alphanumeric string of the form [CCY]-[INDEX]-[TERM]. CCY, INDEX and TERM must be separated by dashes (-). CCY and INDEX must be among the supported currency and index combinations. TERM must be an integer followed by D, W, M or Y. See Table 19.

  For currencies without available ibor indices:
  An alphanumeric string, matching a benchmark curve set up in the market data configuration in `todaysmarket.xml` Yield curves section.

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

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

  Allowable values: A non-negative integer.

- Calendar: The calendar associated to the settlement lag.

  Allowable values: See Table 17 Calendar.

- IssueDate: The issue date of the security.

  See `Date` in Table 13.

- PriceQuoteMethod [Optional]: The quote method of the bond. Bond price quotes and historical bond prices (stored as "fixings") follow this method. Also, the initial price for bond total return swaps follows this method. Defaults to PerentageOfPar.

  Allowable values: PercentageOfPar or CurrencyPerUnit

- PriceQuoteBaseValue [Optional]: The base value for quote method = CurrencyPerUnit. Bond price quotes, historical bond prices stored as fixings and initial prices in bond total return swaps are divided by this value. Defaults to 1.0.

  Allowable values: Any real number.

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

Listing 53: Bond Data

```
<BondData>
    <IssuerId>Ineos Group Holdings SA</IssuerId>
    <CreditCurveId>ISIN:XS0982710740</CreditCurveId>
    <SecurityId>ISIN:XS0982710740</SecurityId>
    <ReferenceCurveId>EUR-EURIBOR-6M</ReferenceCurveId>
    <SettlementDays>2</SettlementDays>
    <Calendar>TARGET</Calendar>
    <IssueDate>20160203</IssueDate>
    <PriceQuoteMethod>PercentageOfPar</PriceQuoteMethod>
    <PriceQuoteBaseValue>1.0</PriceQuoteBaseValue>
    <LegData>
        <LegType>Fixed</LegType>
        <Payer>false</Payer>
        ...
    </LegData>
</BondData>
```

The bond trade type supports perpetual schedules, i.e. perpetual bonds can be represented by omitting the EndDate in the leg data schedule definition. Only rule based schedules can be used to indicate perpetual schedules.

2.2.41 Bond Position

A bond position represents a position in a weighted basket of underlying bonds.

A bond position can be used both as a stand alone trade type (TradeType: *BondPosition*) or as a trade component (`BondBasketData`) used within the *TotalReturnSwap* (Generic TRS) trade type.

It is set up using an `BondBasketData` block as shown in listing 54. The meanings and allowable values of the elements in the block are as follows:

- Quantity: The number of units of the weighted basket held.
  Allowable values: Any positive real number
- Identifier[Optional]: The identifier of the weighted basket. The Underlying data can be retrieved from the reference data via this identifier, if not given in the trade itself. If the bond basket data is set up in the trade itself in Underlying blocks as in in listing 54, no Identifier is required.
  Allowable values: A string that matches the reference data.
- Underlying[Optional]: One or more underlying descriptions. If bond basket data is set up in the reference data for the given identifier, the underlying data will be populated from there and does not need to be provided in the trade. The weighted basket price is then given by

$$\text{Basket-Price} = \text{Quantity} \times \sum_i \text{Weight}_i \times B_i \times \text{FX}_i$$

  where

  - B_i is the price of the ith Bond in the basket

– FX_i is the FX Spot converting from the currency of the ith Bond to the return currency if the BondPosition is in a TotalReturnSwap, otherwise to the currency of the first Bond in the basket.

Allowable values: See 2.3.29 for the definition of an underlying. Only underlyings of Type *Bond* are allowed.

Listing 54: Bond position data

```
<Trade id="BondPosition">
  <TradeType>BondPosition</TradeType>
  <Envelope>...</Envelope>
  <BondBasketData>
    <Quantity>1000</Quantity>
    <Identifier>ISIN:GB00B4KT9Q30</Identifier>
    <Underlying>
      <Type>Bond</Type>
      <Name>US69007TAB08</Name>
      <IdentifierType>ISIN</IdentifierType>
      <Weight>0.5</Weight>
      <BidAskAdjustment>-0.0025</BidAskAdjustment>
    </Underlying>
    <Underlying>
      <Type>Bond</Type>
      <Name>US750236AW16</Name>
      <IdentifierType>ISIN</IdentifierType>
      <Weight>0.5</Weight>
      <BidAskAdjustment>-0.005</BidAskAdjustment>
    </Underlying>
  </BondBasketData>
</Trade>
```

2.2.42 Forward Bond

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

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

Listing 55 shows an example for a physically settled forward bond. Listing 56 shows an example for a cash settled T-Lock transaction specified by a lock-in yield.

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

- BondData: A `BondData` block specifying the underlying bond as described in section 2.2.40. A long position must be taken in the bond, i.e. (`Payer`) flag must be set to (`true`). The bond data block contains additional fields for forward bonds
  - IncomeCurveId: The benchmark curve to be used for compounding, this

must match a name of a curve in the yield curves or index curve block in `todaysmarket.xml`. It is optional to provide this curve. If left out the market reference yield curve from `todaysmarket.xml` is used for compounding.

- SettlementData: The entity defining the terms of settlement:
  - ForwardMaturityDate: The date of maturity of the forward contract.
    Allowable values: See `Date` in Table 13.
  - Settlement [Optional]: Cash or Physical. Option, defaults to Physcial, except in case the settlement is defined by LockRate, in which case it defaults to Cash.
    Allowable values: Cash, Physical
  - Amount [Optional]: The settlement amount (also called strike) transferred at forward maturity in return for the bond (physical delivery) or a cash amount equal to the dirty price of the bond (cash settlement). This is transferred from the party that is long to the party that is short (determined by `LongInForward`) and cannot be a negative amount. It is assumed to be in the same currency as the underlying bond. Exactly one of the fields Amount, LockRate must be given.
    Allowable values: Any non-negative real number.
  - LockRate [Optional]: The payoff is given by (yield at forward maturity - LockRate) x DV01 (LongInForward = true). Exactly one of the fields Amount, LockRate must be given. In case the LockRate is given, the Settlement must be set to Cash. If Settlement is not given, it defaults to Cash in this case.
    Allowable values: Any non-negative real number.
  - LockRateDayCounter [Optional]: The day counter w.r.t. which the lock rate is expressed. Optional, defaults to A360.
    Allowable values: see table 18
  - SettlementDirty [Optional]: A flag that determines whether the settlement amount (`Amount`) reflects a clean (*false*) or dirty (*true*) price. In either case, the dirty amount is actually paid on the forward maturity date, i.e. if SettlementDirty = *false*, the (forward) accruals are computed internally and added to the given amount to get the actual settlement amount. Optional, defaults to true.
    Allowable values: *true*, *false*
- PremiumData: The entity defining the terms of a potential premium payment. This node is optional. If left out it is assumed that no premium is paid.
  - Date: The date when a premium is paid.
    Allowable values: See `Date` in Table 13.
  - Amount: The amount transferred as a premium. This is transferred from the party that is long to the party that is short (determined by `LongInForward`) and cannot be a negative amount. It is assumed to be in the same currency as the underlying bond.

Allowable values: Any non-negative real number.

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

Listing 55: Forward Bond Data

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

Listing 56: Forward Bond Date (T-Lock)

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

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

Forward Bond - Pricing Engine configuration

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

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

2.2.43 Bond Forward / T-Lock / J-Lock (using ref. data)

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

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

Listing 57 shows an example for a physically settled forward bond. Listing 58 shows an example for a cash settled T-Lock transaction specified by a lock-in yield.

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

- The `BondData` block specifies the underlying bond, see below for more details.
  - SecurityId: The underlying security identifier
    Allowable values: Typically the ISIN of the underlying bond, with the ISIN: prefix.
  - BondNotional: The notional of the underlying bond on which the forward is written expressed in the currency of the bond
    Allowable values: Any positive real number.
  - CreditRisk [Optional] Boolean flag indicating whether to show Credit Risk on the Bond product. If set to *false*, the product class will be set to *RatesFX* instead of *Credit*, and there will be no credit sensitivities. Note that if the underlying bond reference is set up without a CreditCurveId - typically for some highly rated government bonds - the CreditRisk flag will have no impact on the product class and no credit sensitivities will be shown even if CreditRisk is set to *true*.
    Allowable Values: *true* or *false* Defaults to *true* if left blank or omitted.
- SettlementData: The entity defining the terms of settlement:
  - ForwardMaturityDate: The date of maturity of the forward contract.
    Allowable values: See `Date` in Table 13.
  - ForwardSettlementDate [Optional]: Settlement date for forward bond or cash settlement payment date.
    Allowable values: See `Date` in Table 13.

- Settlement [Optional]: Cash or Physical. Option, defaults to Physcial, except in case the settlement is defined by LockRate, in which case it defaults to Cash.
  Allowable values: Cash, Physical
- Amount [Optional]: The settlement amount (also called strike) transferred at forward maturity in return for either:
  (a) the bond (physical delivery) or
  (b) a cash amount equal to the dirty price of the bond (cash settlement).
  This is transferred from the party that is long to the party that is short (determined by `LongInForward`) and cannot be a negative amount. It is assumed to be in the same currency as the underlying bond. Exactly one of the fields Amount, LockRate must be given.
  Allowable values: Any non-negative real number.
- LockRate [Optional]: The payoff is given by (yield at forward maturity - LockRate) x DV01 (LongInForward = true). Exactly one of the fields Amount, LockRate must be given. In case the LockRate is given, the Settlement must be set to Cash. If Settlement is not given, it defaults to Cash in this case.
  Allowable values: Any non-negative real number. The LockRate is expressed in decimal form, eg 0.05 is a rate of 5%
- dv01 [Optional]: When the LockRate is given, it is possible to implement a contractual DV01 instead of deriving it from the bond price.
  Allowable values: Any positive real number. E.G If the dPdY is given then dv01=10000*dPdY/N.
- LockRateDayCounter [Optional]: The day counter w.r.t. which the lock rate is expressed. Optional, defaults to A360.
  Allowable values: see table 18
- SettlementDirty [Optional]: A flag that determines whether the settlement amount (`Amount`) reflects a clean (*false*) or dirty (*true*) price. In either case, the dirty amount is actually paid on the forward maturity date, i.e. if SettlementDirty = *false*, the (forward) accruals are computed internally and added to the given amount to get the actual settlement amount. Optional, defaults to true.
  Allowable values: *true*, *false*

- PremiumData: The entity defining the terms of a potential premium payment. This node is optional. If left out it is assumed that no premium is paid.
  - Date: The date when a premium is paid.
    Allowable values: See `Date` in Table 13.
  - Amount: The amount transferred as a premium. This is transferred from the party that is long to the party that is short (determined by `LongInForward`) and cannot be a negative amount. It is assumed to be in the same currency as the underlying bond.
    Allowable values: Any non-negative real number.
- LongInForward: A flag that determines whether the forward contract is entered

in long (*true*) or short (*false*) position.
Allowable values: *true*, *false*

Listing 57: Forward Bond Data

```
<ForwardBondData>
  <BondData>
    <SecurityId>ISIN:XS1234567890</SecurityId>
    <BondNotional>100000</BondNotional>
  <BondData>
  <SettlementData>
    <ForwardMaturityDate>20160808</ForwardMaturityDate>
    <Settlement>Physcial</Settlement>
    <ForwardSettlementDate>20160810</ForwardSettlementDate>
    <Amount>1000000.00</Amount>
    <SettlementDirty>true</SettlementDirty>
  </SettlementData>
  <PremiumData>
    <Amount>1000.00</Amount>
    <Date>20160808</Date>
  </PremiumData>
  <LongInForward>true</LongInForward>
</ForwardBondData>
```

Listing 58: Forward Bond Date (T-Lock)

```
<ForwardBondData>
  <BondData>
    <SecurityId>ISIN:XS1234567890</SecurityId>
    <BondNotional>100000</BondNotional>
  </BondData>
  <SettlementData>
    <ForwardMaturityDate>20160808</ForwardMaturityDate>
    <ForwardSettlementDate>20160810</ForwardSettlementDate>
    <LockRate>0.02365</LockRate>
  </SettlementData>
  <LongInForward>true</LongInForward>
</ForwardBondData>
```

Listing 59: Forward Bond Date (T-Lock) with DV01

```
<ForwardBondData>
<BondData>
  <SecurityId>ISIN:XS1234567890</SecurityId>
  <BondNotional>100000</BondNotional>
</BondData>
<SettlementData>
  <ForwardMaturityDate>20160808</ForwardMaturityDate>
  <ForwardSettlementDate>20160810</ForwardSettlementDate>
  <LockRate>0.02365</LockRate>
  <dv01>0.8</dv01>
</SettlementData>
<LongInForward>true</LongInForward>
</ForwardBondData>
```

2.2.44 Bond Repo

In a bond repo transaction one party A receives a cash amount from a party B for a specified period. At the maturity of the trade party A pays back the cash amount plus accrued interest to party B. Intermediate interest payments are also possible. Party A delivers a bond to party B as a collateral for the received cash amount for the duration of the trade. In exchange the interest to be paid by party A will be lower than for an uncollateralised borrowing transaction.

A bond repo trade is set up using the trade type `BondRepo` and a `BondRepoData` block as shown in listing 60. The block contains two nodes

- `BondData`, which specifies the underlying bond and its quantity, and
- `RepoData`, which specifies the cash leg of the repo

The `BondData` block contains the following fields

- SecurityId: The identified of the underlying security.
  Allowable values: A valid key, usually of the form "ISIN::XY012345679"
- BondNotional: The notional of the underlying bond. This is the effective notional used as collateral, i.e. it should include hair cuts. Usually the number Bond Notional x Bond Dirty Price x (1 - Haircut) will correspond to the nominal on the cash leg at trade inception.
  Allowable values: Any positive real number.
- CreditRisk [Optional] Boolean flag indicating whether to show Credit Risk on the Bond product.
  Allowable Values: *true* or *false* Defaults to *true* if left blank or omitted.

In this case the details of the underlying bond is read from the reference data. It is also possible to inline the details in the trade, see 2.2.40 for more details on this.

The `RepoData` block contains exactly one `LegData` subnode that describes the payments on the cash leg of the repo, see 2.3.3 for details on how to set this up. The `Payer` leg determines whether interest is paid (regular repo) or received (reversed repo).

Listing 60: Bond Repo Data

```
<BondRepoData>
  <BondData>
    <SecurityId>ISIN:US912828X703</SecurityId>
    <BondNotional>27807597.777444</BondNotional>
  </BondData>
  <RepoData>
    <LegData>
      <LegType>Fixed</LegType>
      <Payer>true</Payer>
      <Currency>USD</Currency>
      <Notionals>
        <Notional>28371510.00</Notional>
      </Notionals>
      <ScheduleData>
        <Rules>
          <StartDate>2020-01-06</StartDate>
          <EndDate>2020-04-07</EndDate>
          <Tenor>1Y</Tenor>
          <Calendar>US</Calendar>
          <Convention>MF</Convention>
          <TermConvention>MF</TermConvention>
          <Rule>Forward</Rule>
          <EndOfMonth/>
          <FirstDate/>
          <LastDate/>
        </Rules>
      </ScheduleData>
      <DayCounter>A360</DayCounter>
      <PaymentConvention>F</PaymentConvention>
      <FixedLegData>
        <Rates>
          <Rate>0.0178</Rate>
        </Rates>
      </FixedLegData>
    </LegData>
  </RepoData>
</BondRepoData>
```

2.2.45 Bond Option

A bond option provides the buyer with the right, but not the obligation, to buy or sell a given bond at a fixed price either at or before a specific date. Options are written on government bonds and are traded on an OTC basis.

The structure of a trade node representing a *BondOption* is shown in listing 61:

- The `BondOptionData` node is the trade data container for the option part of a bond option trade type. Vanilla bond options are supported, the exercise style must be *European*. The `BondOptionData` node includes one and only one `OptionData` trade component sub-node plus elements specific to the bond option.
- The latter also includes the underlying Bond description in the `BondData` node, see section 2.2.40, listing 53 for details

Listing 61: Bond Option data

```
<Trade id="...">
  <TradeType>BondOption</TradeType>
  <Envelope>
      ...
  </Envelope>
  <BondOptionData>
    <OptionData>
        ...
    </OptionData>
    <StrikeData>
      <StrikePrice>
        <Value>11809123.56</Value>
        <Currency>EUR</Currency>
      </StrikePrice>
    </StrikeData>
    <Redemption>100.00</Redemption>
    <PriceType>Dirty</PriceType>
    <KnocksOut>false</KnocksOut>
    <BondData>
       <VolatilityCurveId>YieldVols-EUR</VolatilityCurveId>
        ...
    <BondData>
  </BondOptionData>
</Trade>
```

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

- OptionData: This is a trade component sub-node outlined in section 2.3.1 Option Data. Note that the bond option type allows for *European* option style only.
- StrikeData: A node containing the strike information. Allowable values: Supports `StrikePrice` and `StrikeYield` as described in Section 2.3.30.
- Redemption: Redemption ratio in percent
- PriceType: This node defines which strike should be used for the pricing. If the node takes the value Dirty, the strike price should be set equal to the value of the Strike node. If the node takes the value Clean, the strike price should be set equal to the value of the Strike node plus accrued interest at the expiration date of the option.
  Allowable values: Dirty or Clean.
- KnocksOut: If true the option knocks out if the underlying defaults before the option expiry, if false the option is written on the recovery value in case of a default of the bond before the option expiry

The meanings and allowable values of the elements in the `BondData` are:

- VolatilityCurveId: The yield volatility curve to use for the valuation of this bond option.

2.2.46 Bond Option (using bond reference data)

The structure of a trade node representing a *BondOption* is shown in listing 62:

- The `BondOptionData` node is the trade data container for the option part of a bond option trade type. Vanilla bond options are supported, the exercise style must be *European*. The `BondOptionData` node includes one and only one `OptionData` trade component sub-node plus elements specific to the bond option.
- The latter also includes the underlying Bond description in the `BondData` node, see below for details

Note that only par redemption vanilla bonds are supported.

Listing 62: Bond Option data using bond reference data

```
<Trade id="...">
  <TradeType>BondOption</TradeType>
  <Envelope>
      ...
  </Envelope>
  <BondOptionData>
    <OptionData>
     <LongShort>Long</LongShort>
     <OptionType>Call</OptionType>
     <Style>European</Style>
     <ExerciseDates>
      <ExerciseDate>20210203</ExerciseDate>
     </ExerciseDates>
        ...
    </OptionData>
    <StrikeData>
      <StrikePrice>
        <Value>1.23</Value>
      </StrikePrice>
    </StrikeData>
    <PriceType>Dirty</PriceType>
    <KnocksOut>false</KnocksOut>
    <BondData>
       <SecurityId>ISIN:XS1234567890</SecurityId>
       <BondNotional>100000</BondNotional>
    <BondData>
  </BondOptionData>
</Trade>
```

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

- OptionData: This is a trade component sub-node outlined in section 2.3.1 Option Data.

  The relevant fields in the `OptionData` node for a BondOption are:

  - `LongShort` The allowable values are *Long* or *Short*.
  - `OptionType` The allowable values are *Call* or *Put*. For option type *Call*, the Bond Option holder has the right to buy the underlying Bond at the strike

price. For option type *Put*, the Bond Option holder has the right to sell the underlying Bond at the strike price.

  - `Style` The allowable value is *European* only.
  - `Settlement` [Optional] The allowable values are *Cash* or *Physical*, but this field is currently ignored.
  - An `ExerciseDates` node where exactly one ExerciseDate date element must be given.
  - Premiums [Optional]: Option premium amounts paid by the option buyer to the option seller.

    Allowable values: See section 2.3.2

- `StrikeData`: A `StrikeData` node is used as described in Section 2.3.30 to represent the Bond Option strike price or strike yield. If StrikePrice is used, the strike price (`Value` field) is expressed per unit notional, i.e. a strike of 101% of the bond notional is expressed as 1.01. If StrikeYield is used, the `Yield` is quoted in decimal form, e.g. 5% should be entered as 0.05.
- PriceType [Mandatory for StrikePrice, no impact for StrikeYield]:
  The payoff for a bond option is

  max(B - X, 0)

  where B is always the dirty NPV of the underlying bond on the exercise settlement date.
  If `PriceType` is *Clean*, X is (Strike + Underlying Bond Accruals) x BondNotional
  If `PriceType` is *Dirty*, X is Strike x BondNotional

  Allowable values: *Dirty* or *Clean*. If the `StrikeData` node uses StrikeYield, `PriceType` can be omitted as it is not relevant in the yield case.
- KnocksOut: If *true* the option knocks out if the underlying defaults before the option expiry, if *false* the option is written on the recovery value in case of a default of the bond before the option expiry.

  Allowable values: Boolean node, allowing *Y, N, 1, 0, true, false* etc. The full set of allowable values is given in Table 29.

The meanings and allowable values of the elements in the `BondData` are:

- SecurityId: The underlying security identifier

  Allowable values: Typically the ISIN of the underlying bond, with the ISIN: prefix.
- BondNotional: The notional of the underlying bond on which the option is written expressed in the currency of the bond.

  Allowable values: Any positive real number.
- CreditRisk [Optional] Boolean flag indicating whether to show Credit Risk on the Bond product.

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

2.2.47 Bond Total Return Swap

Payoff

A total return swap is a derivative contract in which one counterparty (short) pays out the total returns of an underlying asset and receives a regular fixed or floating cash flow from the other counterparty (long). Here we describe total return swaps with an underlying bond. The total return of the bond is comprised of

- coupon, redemption and amortization payments of the bond, including recovery payments in case of default
- compensation payments that reflect changes of the clean bond value along the TRS schedule.

Input

A vanilla Bond Total Return Swap (Trade type: *BondTRS*) is set up using a `BondTRSData` block as shown in listing 64. The block is comprised of three sub-blocks, which are `BondData`, `TotalReturnData` and `FundingData`.

- The `BondData` block specifies the underlying bond, usually by specifiying the security id and the quantity / bond notional and relying on reference data:
  - SecurityId: The underlying security identifier
    Allowable values: Typically the ISIN of the underlying bond, with the ISIN: prefix.
  - BondNotional: The quantity or number of bonds that is relevant for the TRS, with the convention that 1 bond always corresponds to a face value of 1 unit of bond currency.
    Allowable values: Any positive real number.
  - CreditRisk [Optional] Boolean flag indicating whether to show Credit Risk on the Bond product. If set to *true*, the product class will be set to *Credit* instead of *RatesFX*, and there will be credit sensitivities. Note that if the underlying bond reference is set up without a CreditCurveId - typically for some highly rated government bonds - the CreditRisk flag will have no impact on the product class and no credit sensitivities will be shown even if CreditRisk is set to *true*.
    Allowable Values: *true* or *false* Defaults to *true* if left blank or omitted.

  Alternatively, the BondData block can be specified fully explicit, as outlined in 2.2.40
- The `TotalReturnData` block specifies
  - Payer: Indicates whether the total return leg is paid.
    Allowable values: *true* or *false*
  - InitialPrice [Optional]: Should be filled if the bond price on the first date of the total return schedule is contractually given, in which case the price must correspond to the price type of the total return leg, i.e. if the price type is *Dirty* then the InitialPrice must also be a dirty price (as it is usually given in the term sheet in this case). The price must given in percent, e.g.

101.20.[4] If not given, the bond price for the first date of the total return schedule is read from the price history. Notice that if a bond is quoted in Currency per Unit the initial price should be given in this format too: If e.g. one unit is 50.0 USD an initial price of 51.0 would correspond a dirty amount of 51.0 USD for one unit of the bond.
Allowable values: Any positive real number.

- PriceType: The price type on which these payments are based
  Allowable values: *Dirty* or *Clean*
- ObservationLag [Optional]: The lag between the valuation date and the reference schedule period start date.

  Allowable values: Any valid period, i.e. a non-negative whole number, followed by *D* (days), *W* (weeks), *M* (months), *Y* (years). Defaults to *0D* if left blank or omitted.
- ObservationConvention [Optional]: The roll convention to be used when applying the observation lag.

  Allowable values: A valid roll convention (*F, MF, P, MP, U, NEAREST*), see Table 14 Roll Convention. Defaults to *U* if left blank or omitted.
- ObservationCalendar [Optional]: The calendar to be used when applying the observation lag.

  Allowable values: Any valid calendar, see Table 17 Calendar. Defaults to the *NullCalendar* (no holidays) if left blank or omitted.
- PaymentLag [Optional]: The lag between the reference schedule period end date and the payment date.

  Allowable values: Any valid period, i.e. a non-negative whole number, optionally followed by *D* (days), *W* (weeks), *M* (months), *Y* (years). Defaults to *0D* if left blank or omitted. If a whole number is given and no letter, it is assumed that it is a number of *D* (days).
- PaymentConvention [Optional]: The business day convention to be used when applying the payment lag.

  Allowable values: A valid roll convention (*F, MF, P, MP, U, NEAREST*), see Table 14 Roll Convention. Defaults to *U* if left blank or omitted.
- PaymentCalendar [Optional]: The calendar to be used when applying the payment lag.

  Allowable values: Any valid calendar, see Table 17 Calendar. Defaults to the *NullCalendar* (no holidays) if left blank or omitted.
- PaymentDates [Optional]: This node allows for the specification of a list of explicit payment dates, using `PaymentDate` elements. The list must contain exactly $n - 1$ dates where n is the number of dates in the reference schedule

[4] as opposed to the bond price in the fixing history, where it must be given as 1.0120 and is always a clean quotation

given in the ScheduleData node. See Listing 63 for an example with an assumed ScheduleData with 4 dates.

Listing 63: Payment dates

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

  - FXTerms [Mandatory when underlying bond and BondTRS currencies differ]: Required if the bond currency is different from the return currency, which is always assumed to be equal to the funding leg currency. This kind of trade is also known as a "composite trs". The subnode for the FXTerms node is:
    - FXIndex: The fx index to use for the conversion, this must contain the bond currency and the funding leg currency (in the order defined in table 21, i.e. it does not matter which one is the bond currency and which is the funding currency)

      Allowable values: See Table 21
  - ScheduleData: The reference schedule for the return leg, where the valuation dates are derived from this schedule using the ObservationLag, ObservationConvention and ObservationCalendar fields. The payment dates are derived from this schedule using the PaymentLag, PaymentConvention and PaymentCalendar fields. The payment dates can also be given as an explicit list in the PaymentDates node. Allowable values: A `ScheduleData` block as defined in section 2.3.4
  - PayBondCashFlowsImmediately [Optional]: If true, bond cashflows like coupon or amortisation payments are paid when they occur. If false, these cashflows are paid together with the next return payment. If omitted, the default value is false.

    Allowable values: *true* (immediate payment of bond cashflows) or *false* (bond cashflows are paid on the next return payment date)

- The `FundingData` block specifies the funding leg, which can be of any leg type. The `FundingData` contains exactly one `Leg`. The currency of this leg also defines the currency in which the return is paid. Usually the funding leg's notional will be aligned with the return leg's notional. To achieve this, indexings on the floating leg can be used, see 2.3.8. In the context of bond total return swaps, the indexings can be defined in a simplified way by adding an Indexings node with a subnode FromAssetLeg set to true to the funding leg's LegData node. The notionals node is not required either in the funding leg's LegData in this case. An example for this setup is shown in 64.

Listing 64: Bond Total Return Swap Data with indexed funding leg

```
<BondTRSData>
  <BondData>
    <SecurityId>ISIN:NZIIBDT005C5</SecurityId>
    <BondNotional>100000</BondNotional>
  </BondData>
  <TotalReturnData>
    <Payer>false</Payer>
    <InitialPrice>102.0</InitialPrice>
    <PriceType>Clean</PriceType>
    <ObservationLag>0D</ObservationLag>
    <ObservationConvention>P</ObservationConvention>
    <ObservationCalendar>USD</ObservationCalendar>
    <PaymentLag>2D</PaymentLag>
    <PaymentConvention>F</PaymentConvention>
    <PaymentCalendar>TARGET</PaymentCalendar>
    <!-- <PaymentDates> -->
    <!--   <PaymentDate> ... </PaymentDate> -->
    <!--   <PaymentDate> ... </PaymentDate> -->
    <!-- </PaymentDates> -->
    <FXTerms>
      <FXIndex>FX-TR20H-NZD-USD</FXIndex>
    </FXTerms>
    <ScheduleData>
    ...
    </ScheduleData>
    <PayBondCashFlowsImmediately>false</PayBondCashFlowsImmediately>
  </TotalReturnData>
  <FundingData>
    <LegData>
      <Payer>true</Payer>
      <LegType>Floating</LegType>
      <Currency>USD</Currency>
      ...
      <!-- Notionals node is not required, set to 1 internally -->
      ...
      <Indexings>
      <!-- derive the indexing information (bond price, FX) from the total return leg -->
      <FromAssetLeg>true</FromAssetLeg>
      </Indexings>
      ...
    </LegData>
  </FundingData>
</BondTRSData>
```

2.2.48 Convertible Bond

Payoff

A convertible bond is a bond, that can be converted to a prespecified number of shares. The shares are usually from the bond issuer, but it is also possible that the shares are from a different issuer (exchangeables). In addition, the share currency can be different from the bond currency in both cases (cross-currency convertibles).

The bond might be callable by the issuer (typically in American style) and / or puttable by the investor (typcially in Bermudan style). The issuer calls can be "hard

calls", which are call rights in the traditional sense, as opposed to "soft calls" which can only the exercised if the equity price observed on (and possibly during a period before) the exercise date is above a prespecified threshold. If a soft call is exercised, the investor has the right to convert the bond into shares instead of accepting the payment from the issuer call ("forced conversion").

For a detachable or stripped convertible bond the optionality can be traded separately from the bond. We set the NPV for a detachable convertible bond to the difference of the convertible bond npv and the bond floor npv, where the bond floor denotes the underyling vanilla bond stripped of any optionality.

Additional features of convertible bonds include dividend protection, contingent conversion, mandatory conversion, conversion ratio resets, make-whole calls, dividend-forfeit clauses, copay clauses.

Refer to [32] and [30] for a deeper dive into the convertible bond universe.

Input

A convertible bond is set up in ORE using a `ConvertibleBondData` block as shown in listing 65. The bond details are read from reference data in this case.

A convertible bond is a bond, that can be converted into a prespecified number of shares, given by:

$$NumberOfShares = \frac{BondNotional}{ConversionRatio}$$

Where the Conversion Ratio is specified in the underlying bond reference data.

The shares are usually from the bond issuer, but it is also possible that the shares are from a different issuer (exchangeables). In addition, the share currency can be different from the bond currency in both cases (cross-currency convertibles).

The bond might be callable by the issuer (typically in American style) and / or puttable by the investor (typically in Bermudan style). The issuer calls can be "hard calls", which are call rights in the traditional sense, as opposed to "soft calls" which can only the exercised if the equity price observed on the exercise date is above a prespecified threshold given by TriggerRatios. If a soft call is exercised, the investor has the right to convert the bond into shares instead of accepting the payment from the issuer call ("forced conversion").

The meanings and allowable values of the elements in the `ConvertibleBondData` block are as follows:

- SecurityId: The underlying security identifier
  Allowable values: Typically the ISIN of the underlying bond, with the ISIN: prefix.
- BondNotional: The notional of the underlying bond expressed in the currency of the bond.
  Allowable values: Any positive real number.
- CreditRisk [Optional] Boolean flag indicating whether to show Credit Risk on the Bond product.
  Allowable Values: *true* or *false* Defaults to *true* if left blank or omitted.

Listing 65: Convertible bond set up using reference data

```
<Trade id="ConvertibleBond">
  <TradeType>ConvertibleBond</TradeType>
  <Envelope>...</Envelope>
  <ConvertibleBondData>
    <BondData>
      <SecurityId>ISIN:XS0451905367</SecurityId>
      <BondNotional>1000000.00</BondNotional>
    </BondData>
  </ConvertibleBondData>
</Trade>
```

Alternatively the bond can be set up with further explicit details using the blocks as shown in listing 66. All fields that are not given in the trade XML are filled up with the information from the reference data if available in the reference data. In other words, if reference data is given, the trade xml can still be used to overwrite the information partially, if this seems appropriate. The meanings and allowable values of the elements in the block are as follows:

- BondData: The vanilla part of the bond, see 2.2.40.
- CallData: The call terms of the bond, as described below. Optional, if not given, no calls are present.
- PutData: The put terms of the bond, as described below. Optional, if not given, no puts are present.
- ConversionData: The conversion terms of the bond, as described below. This node must always be given, even if no conversion rights are present (in which case an empty conversion date list can be used).
- DividendProtectionData: The dividend protection terms of the bond, as described below. Optional, if not given, no dividend prtection is present.
- Detachable: If true, the trade represents the embedded optionality, i.e. the difference between the full convertible bond and the bond floor. Optional, defaults to false.
  Allowable values: true, false

The convertible bond trade type supports perpetual schedules, i.e. perpetual convertible bonds can be represented by omitting the EndDate in the following schedules to indicate perpetual schedules. Only rule based schedules can be used to indicate perpetual schedules.

- BondData / LegData: Omitting the EndDate in this schedule indicates that the underlying bond runs perpetually.
- CallData: Omitting the EndDate in this schedule indicates perpetual call dates. For American call dates, where only two dates have to be specified (start and end date of the american call window), a rule based schedule with Tenor = 0D, Rule = Zero and without EndDate can be used to indicate an end date infinitely far away in the future.
- PutData: Same as CallData.

- ConversionData: Omitting the EndDate in this schedule indicates perpetual conversion rights. For American rights, the same comment as under CallData applies.
- ConversionData / ConversionResets: Omitting the EndDate in this schedule indicates perpetual conversion resets.
- DividendProtectionData: Omitting the EndDate in this schedule indicates a perpetual dividend protection schedule.

Listing 66: Convertible bond set up using the detail blocks

```
<Trade id="ConvertibleBond">
  <TradeType>ConvertibleBond</TradeType>
  <Envelope>...</Envelope>
  <ConvertibleBondData>
    <BondData> ... </BondData>
    <CallData> ... </CallData>
    <PutData> ... </PutData>
    <ConversionData> ... </ConversionData>
    <DividendProtectionData> ... </DividendProtectionData>
    <Detachable>false</Detachable>
  </ConvertibleBondData>
</Trade>
```

Specification of CallData / PutData:

All lists specified in subnodes (except the date list itself of course) can be specified as either an explicit list of values corresponding to the schedule dates list or using the attribute `startDate`. An explicit value list can be shorter than the list of dates, in which case the last value from the list is associated to the remaining dates.

See listings 67,68,69,70,71,72,73 for examples of exercise schedules.

- Styles: A list of the exercise styles. Notice that Bermudan is used to define European exercises as well, namely as a Bermudan exercise with a single exercise date. The attribute `startDate` can be used to specify the list.
  Allowable values: American, Bermudan
- ScheduleData: A schedule of exercise dates (for Bermudan exercises) or start / end dates (for American exercises)
  Allowable values: see 2.3.4.
- Prices: A list of exercise prices in relative terms, i.e. if the price is 1.02 then the amount paid on the exercise is this price times the current notional of the bond (plus accrued interest, if the price type is clean, see below). The attribute `startDate` can be used to specify the list.
  Allowable values: Any positive real number.
- PriceType: A list of the flavour in which the exercise prices are given. The attribute `startDate` can be used to specify the list.
  Allowable values: Clean, Dirty.

- IncludeAccrual: A list of flags specifying whether accruals have to be paid on exercise. This is independent of the quoting style of the exercise prices (PriceType).
  Allowable values: true, false

- Soft: A list of flags specifying whether the call is soft (true) or hard (false). The attribute `startDate` can be used to specify the list. Optional, defaults to false. Only applicable to Calls, not to Puts. Optional, if not given, false is assumed, i.e. hard calls. If soft calls are specified, at least one conversion exercise date with corresponding conversion rate must be defined under ConversionData.
  Allowable values: true, false

- TriggerRatios: A list of trigger ratios T for soft calls. A soft call can be executed only if the equity price on the exercise date is above the Conversion Price (defined below) times the trigger ratio, i.e. $S_t > C_t^P T$. Only applicable to Calls, not to Puts. Required for soft calls, can be omitted otherwise.

  $$ConversionPrice, C_t^P = \frac{1}{ConversionRatio}$$

  For cross-currency trades the conversion price is usually quoted in equity ccy, i.e.

  $$ConversionPrice, C_t^P = \frac{1}{ConversionRatio \cdot X_t}$$

  where X_t converts one equity ccy unit to bond ccy

  Allowable values: Any positive real number.

- NOfMTriggers: A list of n-of-m trigger specifications for calls, i.e. the soft-call trigger defined by TriggerRatios must be observed on n of the m calendar days in the period before (and including) a call date. Only applicable to Calls, not to Puts. Optional, defaults to "1-of-1"
  Allowable values: x-of-y with x, y non-negative integers, "1-of-1" corresponds to a vanilla call specification

- MakeWhole: A list of make whole conditions. Optional. Possible subnodes are:

  – ConversionRatioIncrease: In case of a call exercise, the conversion ratio (applicable in case of a forced conversion) is adjusted upwards. The adjustment is additive, i.e. if the current conversion ratio is CR the conversion ratio applicable in case of a forced conversion will be $CR + d$ where d is interpolated from a matrix of effective dates (rows) and stock prices (columns). The conversion rate adjustment might be capped by a prespecified rate. If the exercise date / stock price lies outside the matrix, d is zero, i.e. no adjustment is made. Notice that a soft call trigger is checked w.r.t. CR, i.e. the unadjusted conversion ratio.

    * Cap: An upper bound for the adjusted conversion ratio. Optional, if not given, no cap will be applied.
      Allowable values: Any non-negative real number.

    * StockPrices: A comma separated list of stock prices defining the interpolation grid's x values. At least two stock prices must be given.

Allowable values: A list of non-negative real numbers.

* CrIncreases: A node that contains at least two subnodes CrIncrease. Each subnode must have an attribute startDate defining the effective date of the adjustment and a list of conversion ratio adjustments d. The number of adjustments must match the number of prices given in the StockPrices node.
  Allowable values: A list of non-negative real numbers.

Listing 67: Convertible bond call data example 1

```
<!-- Bermudan issuer call on three dates at a clean price of 100 (hard calls),
     accruals are paid on exercise -->
<CallData>
  <Styles>
    <Style>Bermudan</Style>
  </Styles>
  <ScheduleData>
    <Dates>
      <Dates>
        <Date>2016-08-03</Date>
        <Date>2017-08-03</Date>
        <Date>2018-08-03</Date>
      </Dates>
    </Dates>
  </ScheduleData>
  <Prices>
    <Price>1.00</Price>
  </Prices>
  <PriceTypes>
    <PriceType>Clean</PriceType>
  </PriceTypes>
  <IncludeAccruals>
    <IncludeAccrual>true</IncludeAccrual>
  </IncludeAccruals>
  <Soft>
    <Soft>false</Soft>
  </Soft>
  <TriggerRatios/>
  <NOfMTriggers>
    <NOfMTrigger>20-of-30</NOfMTrigger>
  </NOfMTriggers>
</CallData>
```

Listing 68: Convertible bond call data example 2

```
<!-- Bermudan issuer call on three dates at a clean price of 101, 102 and 103,
     soft calls with trigger ratio of 0.8, 0.85, 0.9,
     accrual are _not_ paid on exercise -->
<CallData>
  <Styles>
    <Style>Bermudan</Style>
  </Styles>
  <ScheduleData>
    <Dates>
      <Dates>
        <Date>2016-08-03</Date>
        <Date>2017-08-03</Date>
        <Date>2018-08-03</Date>
      </Dates>
    </Dates>
  </ScheduleData>
  <Prices>
    <Price>1.01</Price>
    <Price>1.02</Price>
    <Price>1.03</Price>
  </Prices>
  <PriceTypes>
    <PriceType>Clean</PriceType>
  </PriceTypes>
  <IncludeAccruals>
    <IncludeAccrual>false</IncludeAccrual>
  </IncludeAccruals>
  <Soft>
    <Soft>true</Soft>
  </Soft>
  <TriggerRatios>
    <TriggerRatio>0.8</TriggerRatio>
    <TriggerRatio>0.85</TriggerRatio>
    <TriggerRatio>0.9</TriggerRatio>
  </TriggerRatios>
</CallData>
```

Listing 69: Convertible bond call data example 3

```
<!-- American issuer call between 2016-08-03 and 2018-08-03
     at a clean price of 100 (hard calls) -->
<CallData>
  <Styles>
    <Style>American</Style>
  </Styles>
  <ScheduleData>
    <Dates>
      <Dates>
        <Date>2016-08-03</Date>
        <Date>2018-08-03</Date>
      </Dates>
    </Dates>
  </ScheduleData>
  <Prices>
    <Price>1.00</Price>
  </Prices>
  <PriceTypes>
    <PriceType>Clean</PriceType>
  </PriceTypes>
  <IncludeAccruals>
    <IncludeAccrual>true</IncludeAccrual>
  </IncludeAccruals>
  <Soft>
    <Soft>false</Soft>
  </Soft>
  <TriggerRatios/>
</CallData>
```

Listing 70: Convertible bond call data example 4

```
<!-- American issuer call between 2016-08-03 and 2020-08-03 (excl),
     hard calls at 100 between 2016-08-03 and 2018-08-03 (excl),
     soft calls at 102 between 2018-08-03 and 2019-08-03 (excl),
     soft calls at 103 between 2019-08-03 and 2020-08-03 -->
<CallData>
  <Styles>
    <Style>American</Style>
  </Styles>
  <ScheduleData>
    <Dates>
      <Dates>
        <Date>2016-08-03</Date>
        <Date>2018-08-03</Date>
        <Date>2019-08-03</Date>
        <Date>2020-08-03</Date>
      </Dates>
    </Dates>
  </ScheduleData>
  <Prices>
    <Price>1.00</Price>
    <Price startDate="2018-08-03">1.02</Price>
    <Price startDate="2019-08-03">1.03</Price>
  </Prices>
  <PriceTypes>
    <PriceType>Clean</PriceType>
  </PriceTypes>
  <IncludeAccruals>
    <IncludeAccrual>true</IncludeAccrual>
  </IncludeAccruals>
  <Soft>
    <Soft>false</Soft>
    <Soft startDate="2018-03-03">true</Soft>
  </Soft>
  <TriggerRatios>
    <TriggerRatio>0.8</TriggerRatio>
    <TriggerRatio startDate="2019-08-03">0.9</TriggerRatio>
  </TriggerRatios>
</CallData>
```

Listing 71: Convertible bond call data example 5

```
<!-- Bermudan (hard) calls at 100 at 3 dates from 2016 to 2018,
     followed by American (soft) calls at 102 between 2018 and 2020 -->
<CallData>
  <Styles>
    <Style>Bermudan</Style>
    <Style startDate="2018-08-03">American</Style>
  </Styles>
  <ScheduleData>
    <Dates>
      <Dates>
        <Date>2016-08-03</Date>
        <Date>2017-08-03</Date>
        <Date>2018-08-03</Date>
        <Date>2020-08-03</Date>
      </Dates>
    </Dates>
  </ScheduleData>
  <Prices>
    <Price>1.00</Price>
    <Price startDate="2018-08-03">1.02</Price>
  </Prices>
  <PriceTypes>
    <PriceType>Clean</PriceType>
  </PriceTypes>
  <IncludeAccruals>
    <IncludeAccrual>true</IncludeAccrual>
  </IncludeAccruals>
  <Soft>
    <Soft>false</Soft>
    <Soft startDate="2018-08-03">true</Soft>
  </Soft>
  <TriggerRatios>
    <TriggerRatio>0.8</TriggerRatio>
  </TriggerRatios>
</CallData>
```

Listing 72: Convertible bond put data example 6

```
<!-- Bermudan puts calls at 100, 101, 102 at 3 dates from 2016 to 2018 -->
<PutData>
  <Styles>
    <Style>Bermudan</Style>
  </Styles>
  <ScheduleData>
    <Dates>
      <Dates>
        <Date>2016-08-03</Date>
        <Date>2017-08-03</Date>
        <Date>2018-08-03</Date>
      </Dates>
    </Dates>
  </ScheduleData>
  <Prices>
    <Price>1.00</Price>
    <Price>1.01</Price>
    <Price>1.02</Price>
  </Prices>
  <PriceTypes>
    <PriceType>Clean</PriceType>
  </PriceTypes>
  <IncludeAccruals>
    <IncludeAccrual>true</IncludeAccrual>
  </IncludeAccruals>
</PutData>
```

Listing 73: Convertible bond make whole data (conversion ratio increase)

```
<CallData>
...
   <MakeWhole>
     <ConversionRatioIncrease>
       <Cap>0.0740740</Cap>
       <StockPrices>13.50,15.00,16.20,18.00</StockPrices>
       <CrIncreases>
         <CrIncrease startDate="2020-06-25">0.0123456,0.0107487,0.0097173,0.0084567</CrIncrease>
         <CrIncrease startDate="2021-07-01">0.0123456,0.0096880,0.0086963,0.0075294</CrIncrease>
         <CrIncrease startDate="2022-07-01">0.0123456,0.0083927,0.0074222,0.0063383</CrIncrease>
         <CrIncrease startDate="2023-07-01">0.0123456,0.0069360,0.0058790,0.0048322</CrIncrease>
         <CrIncrease startDate="2024-07-01">0.0123456,0.0054453,0.0040025,0.0028833</CrIncrease>
         <CrIncrease startDate="2025-07-01">0.0123456,0.0049380,0.0000000,0.0000000</CrIncrease>
       </CrIncreases>
     </ConversionRatioIncrease>
   </MakeWhole>
</CallData>
```

Specification of ConversionData:

As in the case of the CallData, all lists can be specified as either an explicit list of values corresponding to the schedule dates list or using the attribute `startDate`. The ConversionRatios element is an expcetion, the given start dates are interpreted independently of these schedule dates.

See listings 74, 75,76,77, 78,79 for examples of conversion schedules.

- Styles: The styles of the conversion rights. Notice that Bermudan is used to define European conversion rights as well, namely as a Bermudan conversion right with a single date. The attribute `startDate` can be used to specify the list. Can be omitted, if no conversion dates are given.
  Allwoable values: American, Bermudan
- ScheduleData: The dates defining when the bond is convertible. For Bermudan exercises, the conversion can be executed on the single dates given in the list. For American exercises, the conversion can be executed between a given start and end date. Can be omitted, if no conversion rights are present.
  Allowable values: see 2.3.4.
- ConversionRatios: A list of conversion ratios C^R. The attribute `startDate` can be used to specify a date from which the ratio is valid. Notice that this date is always interpreted "as is", i.e. it is not mapped onto the next date in the defined schedule. If no startDate is given for a ratio, this ratio is interpreted as the initial ratio.
  Allowable values: Any non-negative real number.
- FixedConversionAmounts: If this node is given, the conversion is specified to be conversion to fixed cash amounts instead of equity. If the cash amount currency is different from the bond currency, the FXIndex node must be given. See 79 for an example. As for ConversionRatios the attribute `startDate` can be used to specify a date from which the amount is valid and this date is interpreted "as is", i.e. not mapped onto the next date in the defined schedule. The nodes
  - ConversionRatios
  - ContingentConversion
  - MandatoryConversion
  - ConversionResets
  - Underlying
  - Exchangeable

  must *not* be given, if this node is present. Furthermore, the following nodes from other sections are not applicable if the conversion is specified to be fixed cash amounts, and must therefore not be given:
  - CallData/Soft
  - CallData/TriggerRatios
  - CallData/NoMTriggers
  - CallData/MakeWhole
  - DividendProtectionData (including all subnodes)
- ContingentConversion: This adds a condition $C_t^R S_t > B$ on the convertibility for the periods defined by the conversion dates. Optional.

  - Observations: A list of observation modes.
    Allowable values: Spot (trigger is checked on the conversion date), StartOfPeriod (trigger is checked on the start of the conversion period defined by the dates list, for American style conversion only)
  - Barriers: A list of barriers B associated to the conversion dates.
    Allowable values: Positive real number or zero (conversion is not made contingent for this date).

- MandatoryConversion: This adds a mandatory conversion obligation at a date greater than all other conversion dates (if any). Optional.
  - Date: The mandatory conversion date.
    Allowable values: Any date not earlier than the last otherwise specified conversion date.
  - Type: The type of the mandatory conversion.
    Allowable values: PEPS
  - PepsData: Details of mandatory conversion type PEPS.
    * UpperBarrier: upper barrier for PEPS payoff.
      Allowable values: A real number.
    * LowerBarrier: lower barrier for PEPS payoff.
      Allowable values: A real number.
    * UpperConversionRatio: conversion ratio for upper barrier in PEPS payoff.
      Allowable values: A real number.
    * LowerConversionRatio: conversion ratio for lower barrier in PEPS payoff.
      Allowable values: A real number.

- ConversionResets: This adds a reset schedule for the conversion rate. If a reset feature is defined, only an initial ConversionRatio can be defined, the future conversion ratios are determined by the resets. The startDate attribute can be used to define references, thresholds, gearings, floors, global floors. Optional.
  - ScheduleData: The conversion reset dates.
    Allowable values: see 2.3.4.
  - References: Whether the initial conversion price C_0^P or the current conversion price C_t^P is the reference for the reset.
    Allowable values: InitialConversionPrice, CurrentConversionPrice
  - Thresholds: The threshold T that triggers a reset ($S_t < TC_0^P$ or $S_t < TC_t^P$, depending on Reference)
    Allowable values: positive number or zero (disables the reset on this date effectively)
  - Gearings: The gearings g for the conversion rate adjustment. Option, defaults to 0 (= no gearing applicable)

Allowable values: positive number or zero (no gearing applicable on this date).

  - Floors: The floors f for the conversion rate adjustment. Optional, defaults to 0 (= no floor applicable)
    Allowable values: positive number or zero (no floor applicable on this date)
  - GlobalFloors: The global floors for the conversion rate adjustment. Option, defaults to 0 (= no global floor applicable)
    Allowable values: positive number or zero (no global floor applicable on this date)

- Underlying: The equity underlying.
  Allwoable values: See 2.3.29, the underlying type must be equity.
- FXIndex: If equity ccy is different from bond ccy, an fx index for the two involved ccy is required.
  Allowable values: The format of the FX Index is"FX-SOURCE-CCY1-CCY2" as described in table 21.
- Exchangeable: Node with data for exchangeables. Option, if omitted, the structure is considered non-exchangeable. Subnodes are:
  - IsExchangeable: indicates whether the convertible bond is exchangeable
    Allowable values: true, false
  - EquityCreditCurve: the credit curve modeling the equity issuer default, required if IsExchangeable is true.
    Allowable values: A valid credit curve identifier, e.g the ISIN of a reference bond with the ISIN: prefix: `ISIN:XXNNNNNNNNN`
  - Secured: Indicates whether the convertible is secured with pledged shares or not. Optional, defaults to false.
    Allowable values: true, false.

Listing 74: Convertible bond conversion example 1

```
<!-- Three conversion dates (Bermudan), conversion ratio is 0.5 -->
  <ConversionData>
    <Styles>
      <Style>Bermudan</Style>
    </Styles>
    <ScheduleData>
      <Dates>
        <Dates>
          <Date>2016-08-03</Date>
          <Date>2017-08-03</Date>
          <Date>2018-08-03</Date>
        </Dates>
      </Dates>
    </ScheduleData>
    <ConversionRatios>
      <ConversionRatio>0.05</ConversionRatio>
    </ConversionRatios>
    <Underlying>
      <Type>Equity</Type>
      <Name>RIC:.ABCD</Name>
    </Underlying>
    <FXIndex>FX-ECB-EUR-USD</FXIndex>
    <Exchangeable>
      <IsExchangeable>true</IsExchangeable>
      <EquityCreditCurve>ISIN:XS0982710740</EquityCreditCurve>
      <Secured>true</Secured>
    </Exchangeable>
  </ConversionData>
```

Listing 75: Convertible bond conversion example 2

```
<!-- American conversion between 2016-08-03 and 2020-08-03, with
     conversion ratio 0.5 for 2016-08-03 through 2018-08-03 (excl) and
     conversion ratio 0.6 for 2018-08-03 through 2020-08-03 -->
  <ConversionData>
    <Styles>
      <Style>American</Style>
    </Styles>
    <ScheduleData>
      <Dates>
        <Dates>
          <Date>2016-08-03</Date>
          <Date>2018-08-03</Date>
          <Date>2020-08-03</Date>
        </Dates>
      </Dates>
    </ScheduleData>
    <ConversionRatios>
      <ConversionRatio>0.05</ConversionRatio>
      <ConversionRatio startDate="2018-08-03">0.06</ConversionRatio>
    </ConversionRatios>
    <Underlying>
      <Type>Equity</Type>
      <Name>RIC:.ABCD</Name>
    </Underlying>
  </ConversionData>
```

Listing 76: Convertible bond conversion example 3

```
<!-- American conversion between 2016-08-03 and 2018-08-03, with conversion
     ratio 0.5, the conversion is contingent on the parity being above 1.3
     on 2016-08-03 for the conversion between 2016-08-03 and 2017-08-03 (excl)
     on 2017-08-03 for the conversion between 2017-08-03 and 2018-08-03 -->
  <ConversionData>
    <Styles>
      <Style>American</Style>
    </Styles>
    <ScheduleData>
      <Dates>
        <Dates>
          <Date>2016-08-03</Date>
          <Date>2017-08-03</Date>
          <Date>2018-08-03</Date>
        </Dates>
      </Dates>
    </ScheduleData>
    <ConversionRatios>
      <ConversionRatio>0.05</ConversionRatio>
    </ConversionRatios>
    <ContingentConversion>
      <Observations>
        <Observation>StartOfPeriod</Observation>
      </Observations>
      <Barriers>
        <Barrier>1.3</Barrier>
      </Barriers>
    </ContingentConversion>
    <Underlying>
      <Type>Equity</Type>
      <Name>RIC:.ABCD</Name>
    </Underlying>
  </ConversionData>
```

Listing 77: Convertible bond conversion example 4

```
<!-- American converion between 2016-08-03 and 2018-08-03 with CR 0.5.
     Mandatory conversion on 2020-08-03:
     LowerConversionRatio applies if stock price < LowerBarrier,
     UpperConversionRatio applies if stock price > UpperBarrier -->
  <ConversionData>
    <Styles>
      <Style>American</Style>
    </Styles>
    <ScheduleData>
      <Dates>
        <Dates>
          <Date>2016-08-03</Date>
          <Date>2018-08-03</Date>
        </Dates>
      </Dates>
    </ScheduleData>
    <ConversionRatios>
      <ConversionRatio>0.05</ConversionRatio>
    </ConversionRatios>
    <MandatoryConversion>
      <Date>2020-08-03</Date>
      <Type>PEPS</Type>
      <PepsData>
        <UpperBarrier>32.5</UpperBarrier>
        <LowerBarrier>20.5</LowerBarrier>
        <UpperConversionRatio>0.08</UpperConversionRatio>
        <LowerConversionRatio>0.03</LowerConversionRatio>
      </PepsData>
    </MandatoryConversion>
    <Underlying>
      <Type>Equity</Type>
      <Name>RIC:.ABCD</Name>
    </Underlying>
  </ConversionData>
```

Listing 78: Convertible bond conversion example 5

```
<!-- American conversion between 2016-08-03 and 2018-08-03 with CR 0.5.
     The conversion ratio is reset on 2016-11-03, 2017-02-03, 2018-05-03
     using T = 0.9, g = 0.8, f = 0.6, F = 0.6. -->
  <ConversionData>
    <Styles>
      <Style>American</Style>
    </Styles>
    <ScheduleData>
      <Dates>
        <Dates>
          <Date>2016-08-03</Date>
          <Date>2018-08-03</Date>
        </Dates>
      </Dates>
    </ScheduleData>
    <ConversionRatios>
      <ConversionRatio>0.05</ConversionRatio>
    </ConversionRatios>
    <ConversionResets>
      <ScheduleData>
        <Dates>
          <Dates>
            <Date>2016-11-03</Date>
            <Date>2017-02-03</Date>
            <Date>2018-05-03</Date>
          </Dates>
        </Dates>
      </ScheduleData>
      <References>
        <Reference>InitialConversionPrice</Reference>
      </References>
      <Thresholds>
        <Threshold>0.9</Threshold>
      </Thresholds>
      <Gearings>
        <Gearing>0.8</Gearing>
      </Gearings>
      <Floors>
        <Floor>0.7</Floor>
      </Floors>
      <GlobalFloors>
        <GlobalFloor>15</GlobalFloor>
      </GlobalFloors>
    </ConversionResets>
    <Underlying>
      <Type>Equity</Type>
      <Name>RIC:.ABCD</Name>
    </Underlying>
  </ConversionData>
```

Listing 79: Convertible bond conversion example 6

```
<!-- American conversion between 2024-08-24 and 2027-05-13, with
     conversion to 0.87 GBP cash for 2024-08-24 through 2024-11-23 (excl) and
     conversion to 0.75 GBP cash for 2024-11-23 through 2027-05-13 -->
 <ConversionData>
   <Styles>
     <Style>American</Style>
   </Styles>
   <ScheduleData>
     <Dates>
       <Dates>
         <Date>2024-08-24</Date>
         <Date>2024-11-23</Date>
         <Date>2027-05-13</Date>
       </Dates>
     </Dates>
   </ScheduleData>
   <FixedAmountConversion>
     <Currency>GBP</Currency>
     <Amounts>
       <Amount>0.87</Amount>
       <Amount startDate="2024-11-24">0.75</Amount>
     </Amounts>
   </FixedAmountConversion>
 </ConversionData>
```

Specification of DividendProtectionData:

As for the CallData, all lists can be specified as either an explicit list of values corresponding to the schedule dates list or using the attribute `startDate`.

See listings 80, 81 for examples of dividend protection schedules.

- ScheduleData: The dates of the dividend protection schedule. The first date marks the date when the dividend protection becomes effective, i.e. dividend payments from this date on are taken into account in conversion ratio adjustments or passthroughs. The second date is then the first date on which the accumulated dividends between the first and second date trigger a conversion ratio reset or passthrough, and similar for all subsequent dates. The last given date is the last date with a conversion ratio reset or passthrough.
  Allowable values: see 2.3.4.
- AdjustmentStyles: Whether the dividend excessing the threshold is passed through or the conversion ratio is adjusted. In both cases, the adjustment can be upwards only or up and down.
  Allwoable values: CrUpOnly, CrUpDown, CrUpOnly2, CrUpDown2, PassThroughUpOnly, PassThroughUpDown
- DividendTypes: Whether the conversion ratio adjustment is calculated in terms of absolute or relative dividends. Does not have an effect for pass through dividends (should be set to Aboslute in this case).
  Allwoable values: Absolute, Relative
- Thresholds: The threshold H. Notice that the threshold applies to each single

period of the dividend protection schedule. If the threshold is e.g. provided on an annual basis in the terms of the convertible bond, but the dividend protection schedule is quarterly, then the threshold in the trade xml should be the annual threshold divided by 4.
Allwoable values: Any non-negativee number.

Listing 80: Convertible bond dividend protection example 1

```
<!-- Divdend protection based on aboslute dividend amounts via adjustment
     of the conversion rate, up-only adjustment. -->
  <DividendProtectionData>
    <ScheduleData>
      <Dates>
        <Dates>
          <Date>2016-08-03</Date>
          <Date>2017-08-03</Date>
          <Date>2018-08-03</Date>
          <Date>2019-08-03</Date>
        </Dates>
      </Dates>
    </ScheduleData>
    <AdjustmentStyles>
      <AdjustmentStyle>CrUpOnly</AdjustmentStyle>
    </AdjustmentStyles>
    <DividendTypes>
      <DividendType>Absolute</DividendType>
    </DividendTypes>
    <Thresholds>
      <Threshold>1.2</Threshold>
    </Thresholds>
  </DividendProtectionData>
```

Listing 81: Convertible bond dividend protection example 2

```
<!-- Dividend protection based on relative dividend amounts via adjustment
     of the conversion rate, up-only adjustment. -->
  <DividendProtectionData>
    <ScheduleData>
      <Dates>
        <Dates>
          <Date>2016-08-03</Date>
          <Date>2017-08-03</Date>
          <Date>2018-08-03</Date>
          <Date>2019-08-03</Date>
        </Dates>
      </Dates>
    </ScheduleData>
    <AdjustmentStyles>
      <AdjustmentStyle>CrUpOnly</AdjustmentStyle>
    </AdjustmentStyles>
    <DividendTypes>
      <DividendType>Relative</DividendType>
    </DividendTypes>
    <Thresholds>
      <Threshold>0.01</Threshold>
    </Thresholds>
  </DividendProtectionData>
```

2.2.49 Ascot

Payoff

An Ascot or a Convertible Bond Option is an American style option to buy back a convertible bond. The buyer of a Call Ascot can exercise the deal and get the underlying bond in exchange for paying the strike.

The payout formula for a Call Ascot is:

$$Payout = \max(0, convertiblePrice - Strike)$$

And for a Put Ascot:

$$Payout = \max(0, Strike - convertiblePrice)$$

where:

$$Strike = bondQuantity \cdot (upfrontPayment + assetLeg - redemptionLeg) - fundingLeg$$

Input

An Ascot is set up using an `AscotData` block as shown in listing 82. The bond details are read from reference data in this case.

Listing 82: Ascot set up using reference data

```
<Trade id="Ascot">
  <TradeType>Ascot</TradeType>
  <Envelope>...</Envelope>
  <AscotData>
    <ConvertibleBondData>
      <BondData>
        <SecurityId>ISIN:XY1000000000</SecurityId>
        <BondNotional>1000000.00</BondNotional>
      </BondData>
    </ConvertibleBondData>
    <OptionData>
     <LongShort>Long</LongShort>
     <OptionType>Call</OptionType>
     <Style>American</Style>
     <Settlement>Physical</Settlement>
     <ExerciseDates>
       <ExerciseDate>2029-02-03</ExerciseDate>
     </ExerciseDates>
    </OptionData>
    <ReferenceSwapData>
      <LegData>
        <LegType>Floating</LegType>
        <Payer>false</Payer>
        ...
      </LegData>
    </ReferenceSwapData>
  <AscotData>
</Trade>
```

The meanings and allowable values of the elements in the block are as follows:

- ConvertibleBondData: This describes the underlying convertible bond, see 2.2.48.
- OptionData: This is a trade component sub-node outlined in section 2.3.1 Option Data. The relevant fields in the `OptionData` node for an Ascot are:
  - `LongShort` The allowable values are *Long* or *Short*. The LongShort flag multiplies the option price with +1 / -1. Call and Put payout formulas above are from the long perspective
  - `OptionType` The allowable values are *Call* or *Put*. See payout formulas above.
  - `Style` The Ascot type allows for *American* option exercise style only.
  - `Settlement` The allowable values are *Cash* or *Physical*.
  - An `ExerciseDates` node where exactly one `ExerciseDate` date element must be given.
  - `Premiums` [Optional]: Option premium amounts paid by the option buyer to the option seller.

    Allowable values: See section 2.3.2

- ReferenceSwapData: Contains a single `LegData` node that describes the trade's reference swap funding leg. The asset leg is implied from the bond data. Payer should always be *false* i.e. the swap is entered from the viewpoint of the asset swap buyer.

2.2.50 Collateral Bond Obligation CBO

Payoff

This section sets out the termsheet details for a Collateral Bond Obligaton (or Cashflow CDO). In the context of ORE the name CBO is in use.

We consider an n tranche CBO. The underlying assets consist of a portfolio of corporate bonds or loans with either amortising or bullet structures. The portfolio can contain fixed or floating rate obligations. Maturities cover a range and do need not coincide. We assume that hazard rate data is available and provided externally. There are n tranches in the deal, notes with attachment point A_k and detachment point D_k.

The deal is assumed to be structured as a cashflow securitisation. Interest and Notional repayments are directed in an order of priority first to the note holder. We assume the available pool for Notional repayments consists of scheduled bond notional repayments and recovery amounts. The pool available for interest payments consists of coupons received on the portfolio during the payment period in question.

Class N notes or equity receive the excess pool coupon available after other items in the interest waterfall are discharged.

A typical Interest and Notional waterfall is given in the following.

Interest Waterfall:

1. Taxes
2. Trustee fees and expenses subject to cap
3. Administration fees and expenses subject to cap
4. Payments for hedge transactions other than early termination
5. Interest and fees under the liquidity facility
6. Senior servicing fee
7. Interest due on Senior notes
8. Redemption of Senior notes if over-collateralisation or interest coverage tests not met (sufficient to ensure tests are met)
9. Interest due on next most Senior notes
10. Redemption of next most Senior notes if over-collateralisation or interest coverage tests not met (sufficient to ensure tests are met)
11. $\vdots$
12. Interest Deflection test

13. Ratings Based Deflection test
14. Trustee fees in excess of the cap
15. Administration Fees
16. Payments of hedge termination fees for early termination
17. Subordinated servicing fee
18. Excess to Equity notes up to an IRR Hurdle
19. Management incentive Fee
20. Excess to the Equity notes.

Principal Waterfall:

1. Unpaid items in the first six items of the Interest Waterfall
2. Unpaid Interest on the Senior notes
3. Redemption of the Senior notes if OC and IC tests not met (sufficient to ensure tests are met)
4. Purchase of additional securities during the re-investment period
5. Redemption of the Senior notes
6. Redemption of the next most Senior notes
7. ⋮
8. Unpaid Subordinated Portfolio servicing fees
9. Redemption of the Equity notes

In summary, Interest after tax and expenses goes first to the Senior note holders and then on down the order of priority and finally to the equity noteholder after ensuring that any tests are satisfied. Notional repayments after expenses go to redemption of the senior notes and finally to the equity note holder.

Input

A Cashflow CDO or Collateral Bond Obligation CBO (trade type *CBO*) can be set up in a short version referencing the underlying CBO structure in a static CBO reference datum or a long version, where the CBO structure is specified explicitly.

The main building block is the `CBOData` block as shown in listing 83. The `CBOData` requires the two components `CBOInvestment` and `CBOStructure`. Where the latter represents the general structure, the former specfies the actual investment. For the short version, the CBO is fully specified using the component `CBOInvestment` only, the component `CBOStructure` can be omitted.

Listing 83 exhibits the long version:

Listing 83: CBO Data

```
<CBOData>
  <CBOInvestment>
    <TrancheName>JuniorNote</TrancheName>
    <Notional>4000000.00</Notional>
    <StructureId>Constellation</StructureId>
  </CBOInvestment>
  <CBOStructure>
    <DayCounter>ACT/ACT</DayCounter>
    <PaymentConvention>F</PaymentConvention>
    <Currency>EUR</Currency>
    <ReinvestmentEndDate>2019-12-31</ReinvestmentEndDate>
    <SeniorFee>0.01</SeniorFee>
    <FeeDayCounter>A365</FeeDayCounter>
    <SubordinatedFee>0.02</SubordinatedFee>
    <EquityKicker>0.25</EquityKicker>
    <BondBasketData>
      ...
    </BondBasketData>
    <CBOTranches>
      ...
    </CBOTranches>
    <ScheduleData>
      ...
    </ScheduleData>
  </CBOStructure>
</CBOData>
```

The meanings of the elements of the `CBOData` node follow below:

- TrancheName: Specifies of which tranche, results are shown in the report files (NPV, Sensitivity, ...). The name needs to match one the names specified in `CBOTranches`.
- Notional: Is the invested amount into the tranche specified above. The value is used to scale the NPV from the general tranche NPV, so it may be different to the face amount specified in `CBOTranches`.
- StructureId: if details of the cbo are read from the reference data, StructureId is used as a key.
- DayCounter: The day count convention of the tranches. Allowable values: See table 18.
- PaymentConvention: The payment convention of the tranches. Allowable values: See Table 14 Roll Convention.
- Currency: Defines the currency of the trade, i.e. the currency of the tranches. Allowable values: See Table 15 `Currency`.
- ReinvestmentEndDate: Defines the end of the reinvestment period. During the reinvestment period, principal proceeds are used to reinvest in eliglible assets rather than to redeem CBO notes. Currently the model cannot handle underlying bonds with full amortisation within the reinvestment period. In case the underlying bonds amortise only parts of their full notional (during that

period), the model will leave outstanding balance constant until the end of the reinvestment period. Therafter the underyling bonds amortises at a higher speed.

- SeniorFee: The fee, expressed as rate, paid before all other obligations, top of the waterfall.
- FeeDayCounter: The day count convention for the fees. Allowable values: See table 18.
- SubordinatedFee: The fee, expressed as rate, paid after all other obligations.
- EquityKicker: Fraction x of the residual payment, that will be split among the senior fee receiver (x) and the equity piece (1-x).
- BondBasketData: All specifications of the underlying bond basket. Uses the sub node BondBasketData as described in section 2.3.33.
- CBOTranches: All required instrument data for the tranches of the CBO. Uses the sub node CBOTranches as described in section 2.3.34.
- ScheduleData: This is a trade component sub-node outlined in section 2.3.4 Schedule Data and Dates.

Listing 84 exhibits the reference data in conjunction with short version of the `CBOData` in listing 85. The element meanings are the same as in the long version.

Listing 84: CboReferenceData

```
<ReferenceDatum id="Constellation">
    <Type>CBO</Type>
    <CboReferenceData>
        <Currency>USD</Currency>
        <DayCounter>A365</DayCounter>
        <PaymentConvention>F</PaymentConvention>
        <SeniorFee>0.001</SeniorFee>
        <FeeDayCounter>A365</FeeDayCounter>
        <SubordinatedFee>0.005</SubordinatedFee>
        <EquityKicker>0.01</EquityKicker>
        <CBOTranches>
            ...
        </CBOTranches>
        <ScheduleData>
            ...
        </ScheduleData>
        <BondBasketData>
            ...
        </BondBasketData>
    </CboReferenceData>
</ReferenceDatum>
```

Listing 85: CBOInvestment

```
<CBOData>
  <CBOInvestment>
    <TrancheName>JuniorNote</TrancheName>
    <Notional>4000000.00</Notional>
    <StructureId>Constellation</StructureId>
  </CBOInvestment>
</CBOData>
```

2.2.51 Composite Trade

A composite trade is a hybrid position consisting of multiple component trades. As such it inherits the characteristics of the trades defined within it. Examples of Composite Trades include combinations of vanilla options like straddles.

The `CompositeTradeData` node is the trade data container for the *CompositeTrade* trade type. A composite trade is a hybrid position consisting of multiple component trades. The structure of an example `CompositeTradeData` node for a commodity option is shown in Listing 86.

Listing 86: Composite trade data

```
<CompositeTradeData>
  <Currency>USD</Currency>
  <NotionalCalculation>Sum</NotionalCalculation>
  <Components>
    <Trade id="">
      <!-- A valid trade xml -->
    </Trade>
    <Trade id="">
      <!-- A valid trade xml -->
    </Trade>
  </Components>
</CompositeTradeData>
```

Listing 87: Composite trade data with Portfolio Reference Data

```
<CompositeTradeData>
  <Currency>USD</Currency>
  <NotionalCalculation>Sum</NotionalCalculation>
  <PortfolioBasket>true</PortfolioBasket>
  <BasketName>NAME</BasketName>
<IndexQuantity>100</IndexQuantity>
</CompositeTradeData>
```

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

- Currency: Defines the currency the NPV of the composite trade will be represented in.
  Allowable values: See Table 15 `Currency`.

- NotionalCalculation [Optional]: The method by which the notional of the composite trade will be calculated.
  Allowable values:

  Sum: The notional will be calculated as the sum of the notionals of the constituent trades. This is the default behaviour if the field is omitted (unless an override is provided).

  Mean or *Average*: The notional will be calculated as the mean of the notionals of the constituent trades.

  First: The notional of the first constituent trade will be used.

  Last: The notional of the first constituent trade will be used.

  Min: The notional will be calculated as the minimum of the notionals of the constituent trades.

  Max: The notional will be calculated as the minimum of the notionals of the constituent trades.

  Override: the notional will be read directly from the notional override field.
- NotionalOverride [Optional]: The notional which will be used for the trade, overriding any calculation method specified.
  Allowable values: Any non-negative real number.
- Components: The portfolio of trades that make up the composite trade.
  Allowable values: These trades should be valid xmls that could otherwise be entered into the portfolio, with the exception that they can have empty ids.
- PortfolioBasket [Optional]: Indicate if the Component represent a portfolio basket.
  Allowable values: Boolean true or false.
- BasketName [Optional]: The portfolio Id.
  Allowable values: Any string. Note that if PortfolioBasket is True then there must be a BasketName. We look up the Basket within the reference data.
- IndexQuantity [Optional]: Number of shares of the index.

2.2.52 Credit Default Swap / Quanto Credit Default Swap

Payoff

A CDS is a credit derivative between two counterparties. The buyer of a CDS makes premium payments to the CDS seller. In return, the seller agrees that in the event that an underlying reference entity defaults or experiences a credit event, seller will compensate the buyer, in relation to a financial instrument issued by the reference entity.

- Reference entity is typically a corporation, government that has issued loans or bonds, and is not party to the CDS contract.
- Single name as well as Indices are supported as reference entities.

In a fixed recovery CDS, the recovery rate that is used when determining the payment on the occurence of a credit event is specified up front in the contract.

Input

A credit default swap, trade type `CreditDefaultSwap`, is set up using a `CreditDefaultSwapData` block as shown in listing 88 or 89. The `CreditDefaultSwapData` block must include either a `CreditCurveId` element or a `ReferenceInformation` node.

The `LegData` sub-node must be a fixed leg, and represents the recurring premium payments. The direction of the fixed leg payments define if the CDS is for bought (`Payer`: *true*) or sold (`Payer`: *false*) protection.

The elements have the following meaning:

- IssuerId [Optional]: An identifier for the reference entity of the CDS. For informational purposes and not used for pricing.
- CreditCurveId: The identifier of the reference entity defining the default curve used for pricing. For the allowable values, see `CreditCurveId` for credit trades - single name in Table 23. A `ReferenceInformation` node may be used in place of this `CreditCurveId` node.
- ReferenceInformation: This node may be used as an alternative to the `CreditCurveId` node to specify the reference entity, tier, currency and documentation clause for the CDS. This in turn defines the credit curve used for pricing. The `ReferenceInformation` node is described in further detail in Section 2.3.27.
- SettlesAccrual [Optional]: Whether or not the accrued coupon is due in the event of a default. This defaults to `true` if not provided.

  Allowable values: Boolean node, allowing *Y, N, 1, 0, true, false* etc. The full set of allowable values is given in Table 29.
- ProtectionPaymentTime [Optional]: Controls the payment time of protection and premium accrual payments in case of a default event. Defaults to `atDefault`.

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

  Allowable values: Boolean node, allowing *Y, N, 1, 0, true, false* etc. The full set of allowable values is given in Table 29.
- ProtectionStart [Optional]: The first date where a credit event will trigger the contract. This defaults to the first date in the schedule if it is not provided. Must be set to a date before or on the first date in the schedule if the `LegData` has a rule that is not one of `CDS` or `CDS2015`. In general, for standard CDS traded after the CDS Big Bang in 2009, the protection start date is equal to the

trade date. Therefore, typically the `ProtectionStart` should be set to the trade date of the CDS.

- UpfrontDate [Optional]: Settlement date for the UpfrontFee if an UpfrontFee is provided. If an UpfrontFee is provided and it is non-zero, `UpfrontDate` is required. The `UpfrontDate`, if provided, must be on or after the ProtectionStart date. Typically, it is 3 business days after the CDS contract trade date.
- UpfrontFee [Optional]: The upfront payment, expressed as a percentage in decimal form, to be multiplied by notional amount. If an UpfrontDate is provided, an UpfrontFee must also be provided. The UpfrontFee can be omitted but cannot be left blank. The UpfrontFee can be negative. The UpfrontFee is treated as an amount payable by the protection buyer to the protection seller. A negative value for the UpfrontFee indicates that the UpfrontFee is being paid by the protection seller to the protection buyer.

  Allowable values: Any real number, expressed in decimal form as a percentage of the notional. E.g. an UpfrontFee of *0.045* and a notional of 10M, would imply an upfront fee amount of 450K.
- FixedRecoveryRate [Optional]: This node holds the fixed recovery rate if the CDS is a fixed recovery CDS. For a standard CDS, this field should be omitted.
- TradeDate [Optional]: The CDS trade date. If omitted, the trade date is deduced from the protection start date. If the schedule provided in the `LegData` has a rule that is either `CDS` or `CDS2015`, the trade date is set equal to the protection start date. This is the standard for CDS traded after the CDS Big Bang in 2009. Otherwise, the trade date is set equal to the protection start date minus 1 day as it was standard before the CDS Big Bang to have protection starting on the day after the trade date.
- CashSettlementDays [Optional]: The number of business days between the trade date and the cash settlement date. For standard CDS, this is 3 business days. If omitted, this defaults to 3.
- RebatesAccrual [Optional]: The protection seller pays the accrued scheduled current coupon at the start of the contract. The rebate date is not provided but computed to be two days after protection start. This defaults to `true` if not provided.

  Allowable values: Boolean node, allowing *Y, N, 1, 0, true, false* etc. The full set of allowable values is given in Table 29.

The `LegData` block then defines the CDS premium leg structure. This premium leg must be be of type `Fixed` as described in Section 2.3.5.

Listing 88: CreditDefaultSwap Data

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

Listing 89: `CreditDefaultSwapData` with `ReferenceInformation`

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

A quanto credit default swap is a credit default swap with different denomination and settlement currencies. Listing 90 shows an Example: The trade has a notional of 50 million BRL and pays a 6% premium. The premuim amounts are converted using the FX-TR20H-USD-BRL fixing two days before they are settled in USD. The hypothetical protection amounts computed for pricing purposes are converted to USD in a similar fashion.

Listing 90: Quanto CDS CreditDefaultSwap Data

```
<LegData>
  <LegType>Fixed</LegType>
  <Payer>true</Payer>
  <!-- This is the settlement currency -->
  <Currency>USD</Currency>
  <!-- This is the BRL notional -->
  <Notionals>
    <Notional>50000000</Notional>
  </Notionals>
  <!-- The FX index used to convert BRL amounts to the settlement ccy USD -->
  <Indexings>
    <Indexing>
      <Index>FX-TR20H-USD-BRL</Index>
      <FixingDays>2</FixingDays>
      <FixingCalendar>USD,BRL</FixingCalendar>
      <IsInArrears>true</IsInArrears>
    </Indexing>
  </Indexings>
  ...
  <FixedLegData>
    <Rates>
      <Rate>0.06</Rate>
    </Rates>
  </FixedLegData>
 ...
</LegData>
```

2.2.53 Index Credit Default Swap

An index credit default swap (trade type *IndexCreditDefaultSwap*) is set up using an `IndexCreditDefaultSwapData` block as shown in listing 91 and includes `LegData` and `BasketData` trade component sub-nodes.

The `LegData` sub-node must be a fixed leg, and represents the recurring premium payments. The direction of the fixed leg payments define if the Index CDS is for bought (`Payer`: *true*) or sold (`Payer`: *false*) protection. The notional on the fixed leg is the "unfactored notional", i.e. the notional excluding any defaults. This is opposed to the "trade date notional" which is reduced by defaults since the series inception until the trade date and the "current notional" or "factored notional" which is reduced by defaults between the series inception and the current evaluation date of the trade.

The `BasketData` sub-node (see section 2.3.28) is optional and specifies the constituent reference entities of the index. This sub-node is intended for non-standard indices, that require a bespoke basket. When `BasketData` is omitted, the index constituents are derived from the `CreditCurveId` element in the `IndexCreditDefaultSwapData` block.

Listing 91: Index CreditDefaultSwap Data

```
<IndexCreditDefaultSwapData>
  <CreditCurveId>RED:2I65BRHH6</CreditCurveId>
  <SettlesAccrual>Y</SettlesAccrual>
  <ProtectionPaymentTime>atDefault</ProtectionPaymentTime>
  <ProtectionStart>20160206</ProtectionStart>
  <UpfrontDate>20160208</UpfrontDate>
  <UpfrontFee>0.0</UpfrontFee>
  <LegData>
        <LegType>Fixed</LegType>
        <Payer>false</Payer>
        ...
  </LegData>
   <BasketData>
    <Name>
      <IssuerId>CPTY_1</IssuerId>
      <CreditCurveId>RED:</CreditCurveId>
      <Notional>100000.0</Notional>
      <Currency>USD</Currency>
    </Name>
    <Name>
      <IssuerId>CPTY_2</IssuerId>
      <CreditCurveId>RED:</CreditCurveId>
      <Notional>100000.0</Notional>
      <Currency>USD</Currency>
    </Name>
    <Name>
      <IssuerId>CPTY_3</IssuerId>
      <CreditCurveId>RED:</CreditCurveId>
      <Notional>100000.0</Notional>
      <Currency>USD</Currency>
    </Name>
    <!-- ... -->
  </BasketData>
</IndexCreditDefaultSwapData>
```

The meanings of the elements of the `IndexCreditDefaultSwapData` node follow below:

- CreditCurveId: The identifier of the index defining the default curve used for pricing. The pricing can be set up to either use the index curve id, or use the curve id:s of the individual index components defined in `BasketData`.

  Allowable values: See `CreditCurveId` for credit trades - index in Table 23. Note that the `CreditCurveId` cannot be a redcode or other identifier for an ABX or CMBX. For these underlyings, trade type *AssetBackedCreditDefaultSwap* is used instead.

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

  Allowable values: Boolean node, allowing *Y, N, 1, 0, true, false* etc. The full set of allowable values is given in Table 29.

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

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

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

  Allowable values: Boolean node, allowing *Y, N, 1, 0, true, false* etc. The full set of allowable values is given in Table 29.

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

  Allowable values: See `Date` in Table 13.

- UpfrontDate [Optional]: Settlement date for the UpfrontFee if an UpfrontFee is provided. If an UpfrontFee is provided and it is non-zero, UpfrontDate is required.

  Allowable values: See `Date` in Table 13. The UpfrontDate, if provided, must be on or after the ProtectionStart date.

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

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

- TradeDate [Optional]: The index CDS trade date. If omitted, the trade date is deduced from the protection start date. If the schedule provided in the `LegData` has a rule that is either `CDS` or `CDS2015`, the trade date is set equal to the protection start date. Otherwise, the trade date is set equal to the protection start date minus 1 day.

  Allowable values: See `Date` in Table 13.

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

  Allowable values: Any non-negative integer.

- RebatesAccrual [Optional]: The protection seller pays the accrued scheduled current coupon at the start of the contract. The rebate date is not provided but

computed to be two days after protection start. This defaults to `true` if not provided.

Allowable values: Boolean node, allowing *Y, N, 1, 0, true, false* etc. The full set of allowable values is given in Table 29.

The `LegData` block then defines the Index CDS premium leg structure. This premium leg must be be of type `Fixed` as described in Section 2.3.5.

2.2.54 Index Credit Default Swap Option

Payoff

A CDS Option is an option to buy or sell protection as a credit default swap on a specific reference credit with a specific maturity. The option is usually European, exercisable only at one date in the future at a specific strike price defined as a coupon rate on the credit default swap.

Credit default options can be options on single name credits or credit indices such as iTraxx or CDX. CDS Options on single credits are extinguished upon default without any cashflows, other than the upfront premium paid by the buyer of the option. Options on credit indices include any defaulted entities in the intrinsic value of the option when exercised.

CDS Options can come with or without *front end protection*, i.e. the protection seller may either have to pay the contract notional if default happens before option expiry, or not (knock out).

Input

An index CDS option has trade type `IndexCreditDefaultSwapOption` in ORE. The Index CDS Option is set up using an `IndexCreditDefaultSwapOptionData` node as shown in Listing 92. Its child nodes have the following meanings:

- `IndexTerm` [Optional]: An optional node giving the term of the underlying index CDS e.g. 3Y, 5Y, 7Y, 10Y etc. The main function of this node is to allow for different index CDS option volatility structures for different terms of the same index series e.g. a CDX HY Series 34 5Y volatility structure and a CDX HY Series 34 10Y volatility structure. If this node is omitted, the market is searched for a CDS volatility surface with ID equal to the value of the `CreditCurveId` node under `IndexCreditDefaultSwapData`. There will generally be one `CreditCurveId` for each index CDS series e.g. `CDXHYS34V1` for CDX HY Series 34 Version 1. Consequently, there can only be one CDS volatility surface for this index CDS series. When `IndexTerm` is populated with the underlying index term, the market is searched for a CDS volatility surface with ID equal to the value of the `CreditCurveId` node with suffix `-[IndexTerm]`. For example, if the `CreditCurveId` node on an index CDS option trade is `CDXHYS34V1` and the `IndexTerm` node is populated with `5Y`, the market will be searched for a CDS volatility surface with ID `CDXHYS34V1-5Y` and this will be used in the trade valuation. In this way, different volatility surfaces can be used to value different terms of the same CDS index series.

  Allowable values: A string that can be parsed as a term that is a valid term for

the underlying CDS index e.g. *5Y*, *10Y*, etc. Defaults to *5Y* if omitted.

- `OptionData`: A node defining the option details as described in Section 2.3.1. The relevant fields in the `OptionData` node for an IndexCDSOption are:
  - `LongShort` The allowable values are *Long* or *Short*. *Long* meaning that the holder has the option to enter into the underlying index CDS.
  - `OptionType` [Optional] *Put/Call* is optional and not used. The `Payer` field in the underlying Index CDS leg determines if the option is to buy or sell protection. The `Payer` field is from the perspective of the party that is long.
  - `Style` Must be set to *European* as this is the only supported exercise for `IndexCreditDefaultSwapOption`.
  - `Settlement` The allowable values are *Cash* or *Physical*.
  - `PayOffAtExpiry` Must be set to *false* as only payoff at exercise is supported.
  - An `ExerciseDates` node where exactly one ExerciseDate date element must be given.
  - `Premiums` [Optional]: Option premium amounts paid by the option buyer (*Long*) to the option seller (*Short*). See section 2.3.2
- `IndexCreditDefaultSwapData`: A node defining the underlying index CDS as described in Section 2.2.53. Note that the `StartDate` in the `Scheduledata` in the premium leg in the `IndexCreditDefaultSwapData` should be the date on which the underlying CDS is entered into if the option is exercised (as opposed to the inception date of the underlying index CDS series). Under standard terms, the `StartDate` would be equal to the `ExerciseDate` but it can also be on a date after the `ExerciseDate`, but not on a date before the `ExerciseDate`, unless Rule is *CDS2015* or *CDS* and `StartDate` is set at the start of the full IMM period that the `ExerciseDate` falls into.

  The `TradeDate` and `ProtectionStart` on the underlying CDS do not need to be populated. If omitted, which is recommended, the `TradeDate` and `ProtectionStart` on the underlying CDS default as follows:

  `TradeDate` = max (option `ExerciseDate`, underlying schedule `StartDate`)
  `ProtectionStart` = max (option `ExerciseDate`, underl. schedule `StartDate`)

  Note that the cash settlement date for the underlying swap upfront premium is set to the underlying `TradeDate` with defaults as above, plus 3 business days.

  Also note that for schedules with IMM rules (e.g. *CDS2015*), if the underlying schedule `StartDate` is not falling on an IMM date, it is adjusted to the previous quarterly IMM date.

  Finally, the notional is - as in the case of an Index Credit Default Swap - the "unfactored notional", i.e. the notional excluding any defaults between the series inception and the trade or evaluation date of the trade.
- `Strike` [Optional]: A real number defining the option strike level. If this is an empty string or omitted the strike will be determined according to table 3.

Note that if a strike is given, the UpfrontFee on the underlying IndexCDS must be zero or omitted. The UpfrontFee is interpreted as a price strike.

Allowable values: Any real number. Note that the `Strike` is expressed in decimal form when `StrikeType` is *Spread*, and in decimal form as percentage of notional when `StrikeType` is *Price*. I.e. a `Strike` of 1.05 is 105% of the notional when `StrikeType` is *Price*.

- `StrikeType` [Optional]: Determines the strike type. If *Spread* is given, the `Strike` is interpreted as a strike *spread*. If *Price* is given, the `Strike` is interpreted as a strike *price*. If omitted or left blank, it will be determined according to table 3.

  Allowable values: *Spread* or *Price*. Note that *Spread* is only supported when the underlying market data is set up with spread strikes, and *Price* is only supported when the market data is set up with price strikes. Typically the market data convention for Index CDS Options is spread strikes, with the exception of CDX North America High Yield (CDX NA HY) names, where the convention is to use price strikes.

- `TradeDate` [Optional]: The trade date. If not given defaults to the valuation date. In case of an underlying default the trade date is used to determine whether the underlying notional before default should be considered part of the outstanding notional (TradeDate $<$ AuctionDate) or not (TradeDate $\geq$ AuctionDate).

  Allowable values: See `Date` in Table 13. Can not be later than the valuation date.

- `FrontEndProtectionStartDate` [Optional]: The date on which the front end protection kicks in. If not given, it defaults to the TradeDate. In case of an underlying default this date is used to determine whether the underlying contributes to the realised front end protection amount (FrontEndProtectionStartDate $<$ AuctionDate) or not (FrontEndProtectionStartDate $\geq$ AcutionDate).

  Allowable values: See `Date` in Table 13. Can not be later than the trade date.

- `FixedRecoveryRate` [Optional]: If provided, this recovery rate will be used in place of the market quoted recovery rate of the underlying.

  Allowable values: Any real number in the range [0,1]. If omitted, the market quoted recovery rate of the underlying is used.

Listing 92: Example Structure of `IndexCreditDefaultSwapOptionData` *node.*

```
<IndexCreditDefaultSwapOptionData>
  <IndexTerm>5Y</IndexTerm>
  <OptionData>
      <LongShort>Long</LongShort>
      <Style>European</Style>
      <Settlement>Cash</Settlement>
      <PayOffAtExpiry>false</PayOffAtExpiry>
      <ExerciseDates>
        <ExerciseDate>2023-05-09</ExerciseDate>
      </ExerciseDates>
  </OptionData>
  <IndexCreditDefaultSwapData>
    ...
  </IndexCreditDefaultSwapData>
  <Strike>1.063</Strike>
  <StrikeType>Price</StrikeType>
</IndexCreditDefaultSwapOptionData>
```

Strike	StrikeType	UpfrontFee	Effective Strike	Effective StrikeType
na	na	na	RunningCoupon	Spread
na	Spread	na	RunningCoupon	Spread
na	Price	na	1.0	Price
K	na	na	K	Spread
K	Spread	na	K	Spread
K	Price	na	K	Price
na	na	U	1.0 - U	Price
na	Spread	U ($=0$)	RunningCoupon	Spread
na	Spread	U ($\neq 0$)	(not allowed)	(not allowed)
na	Price	U	1.0 - U	Price
K	na	U ($=0$)	K	Spread
K	na	U ($\neq 0$)	(not allowed)	(not allowed)
K	Spread	U ($=0$)	K	Spread
K	Spread	U ($\neq 0$)	(not allowed)	(not allowed)
K	Price	U ($=0$)	K	Price
K	Price	U ($\neq 0$)	(not allowed)	(not allowed)

Table 3: Effective strike and strike type to be used in an Index CDS Option dependent on the Strike, StrikeType and UpfrontFee in the underlying Index CDS

2.2.55 Synthetic CDO

Payoff

A synthetic CDO is a basket credit derivative, where the protection seller receives a premium cash flow in exchange for providing (notional) protection against portfolio losses due to defaults in a specific tranche characterized by the attachment point A and detachment point D. Attachment point A is the fraction of the portfolio loss-given-default (LGD) where the protection starts, and the detachment point is the corresponding fraction where protection stops. The LGD is given by the sum of

defaulted asset notional amounts N_i, reduced by their respective recovery rate R_i,i.e. $LGD_i = (1 - R_i)\, N_i$.

CDOs can refer to the constituents of an index such as CDX or iTraxx, or be bespoke, i.e. refer to a bespoke basket of underlying credit names, using the `BasketData` sub-node, (see section 2.3.28)

Input

A Synthetic Collateralized Debt Obligation (CDO), uses trade type *SyntheticCDO* and is set up using a `CdoData` block as shown in listing 93.

Listing 93: CDO Data

```
<CdoData>
  <Qualifier>RED:2I65BRHH6</Qualifier>
  <AttachmentPoint>0.12</AttachmentPoint>
  <DetachmentPoint>0.22</DetachmentPoint>
  <ProtectionStart> 20140425 </ProtectionStart>
  <UpfrontDate/>
  <UpfrontFee/>
  <SettlesAccrual>Y</SettlesAccrual>
  <ProtectionPaymentTime>atDefault</ProtectionPaymentTime>
  <!-- Premium leg -->
  <LegData>
   <LegType>Fixed</LegType>
   <Payer>true</Payer>
      ...
  </LegData>
  <BasketData>
    ...
  </BasketData>
</CdoData>
```

The meanings of the elements of the `CdoData` node follow below:

- Qualifier: The identifier of the credit index defining the default and base correlation curves used for pricing. In the case of a bespoke basket, i.e. when the `BasketData` sub-node is used, the Qualifer should be set to the credit index most closely matching the bespoke basket.

  Allowable values: See `CreditCurveId` for credit trades - index in Table 23.

- AttachmentPoint: Losses where protection starts, expressed as a fraction of the basket notional. Note that Attachment- and DetachmentPoints (AP, DP) are defined as fractions of the current basket notional.

  Allowable values: A number between 0 and 1, below the DetachmentPoint.

- DetachmentPoint: Losses where protection end, expressed as a fraction of the basket notional

  Note that Attachment- and DetachmentPoints (AP, DP) are defined as fractions of the current basket notional.

  Allowable values: A number between 0 and 1, above the AttachmentPoint.

- SettlesAccrual: Whether or not the accrued coupon is due in the event of a default.

  Allowable values: Boolean node, allowing *Y, N, 1, 0, true, false* etc. The full set of allowable values is given in Table 29.

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

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

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

  Allowable values: Boolean node, allowing *Y, N, 1, 0, true, false* etc. The full set of allowable values is given in Table 29.

- ProtectionStart: The first date where a default event will trigger the contract

  Allowable values: See `Date` in Table 13. Must be set to a date before or on the first date in the premium leg schedule.

- UpfrontDate[Optional]: Settlement date for the UpfrontFee if an UpfrontFee is provided. If an UpfrontFee is provided and it is non-zero, UpfrontDate is required.

  Allowable values: See `Date` in Table 13. The UpfrontDate, if provided, must be on or after the ProtectionStart date.

- UpfrontFee[Optional]: The upfront payment, expressed as a rate, to be multiplied by the tranche `Notional` amount. Note that a positive amount indicates that the UpfrontFee is paid by the protection buyer to the protection seller, and a negative amount indicates that the UpfrontFee is paid by the protection seller to the protection buyer. The UpfrontFee cannot be left blank.

  Allowable values: Any real number

- FixedRecoveryRate[Optional]: If provided, this recovery rate will be used in palce of the market quoted recovery rates of the underlying basket or index constituents, to work out the portfolio loss distribution and expected tranche loss.

  Allowable values: Any real number in the range [0, 1]

- LegData: Premium leg description as in an Index CDS (see section 2.2.53) with `Notional` corresponding to the initial tranche notional.
  Note that the `Payer` field in `LegData` determines whereas protection is bought (*true*) or sold (*false*).
  The `StartDate` in `LegData` is the first accrual start date on the premium leg of the index tranche. If the date generation `Rule` is *CDS2015*, one can enter the index tranche trade date for StartDate and the correct accrual start date will be deduced, i.e. the first accrual start date before the trade date using the *CDS2015* date generation rules.

- BasketData[Optional]: Underlying basket description for bespoke baskets (see section 2.3.28). This is analogous to a bespoke basket in an Index CDS (see section 2.2.53). If omitted, CreditIndex static data, with *id* `Qualifier` element in `CdoData`, is extracted from ReferenceData.

  Note that the sum of notionals of the basket components must add up to the complete basket notional.

  $SumofComponentNotionals = CompleteBasketNotional = InitialTrancheNotional/(DetachmentPoint - AttachmentPoint)$

  If weights are used instead of notionals in the basket components, the sum of the weights must add up to 1.

2.2.56 Risk Participation Agreement (RPA)

Payoff

A RPA is a credit derivative between two counterparties. The protection buyer pays a premium to the protection seller. In return the protection seller agrees to pay a percentage share (the "participation rate") of all liabilities of a referenced underlying swap transaction payable by a one of counterparties of that transaction (the "reference entity") in case of the default of this entity. The protection is granted for a defined period of time, which often coincides with the lifetime of the underlying swap. The premium payments may comprise only one upfront payment or a periodic series of payments. In case of a credit event premiums to be paid after the event are cancelled.

In a fixed recovery RPA, the recovery rate that is used to determine the payment on the occurence of a credit event is specified up front in the contract.

Input

A risk participation agreement is set up using the trade type `RiskParticipationAgreement` and a `RiskParticipationAgreementData` block as shown in listing 94. The block contains a `ProtectionFee` block that can include one or more legs representing the fees paid by the protection buyer and an `Underlying` block containing either the legs of the underlying swap or the Treasury-Lock data that the contract references.

If the underlying reference entity defaults, the protection buyer receives the PV of the underlying if this is positive. Here, the underlying PV is computed using the payer / receiver flags as set up for the legs under the underlying node. Whether the trade represents a protection buyer or seller position is indicated by the payer flag in the protection fee leg data: If true protection is bought (and the protection fee is paid), if false the protection is sold (and the protection fee is received).

Listing 94: Risk Participation Agreement Data

```
<RiskParticipationAgreementData>
  <ParticipationRate>0.8</ParticipationRate>
  <ProtectionStart>2018-10-01</ProtectionStart>
  <ProtectionEnd>2038-10-01</ProtectionEnd>
  <CreditCurveId>RED:008CA0|SNRFOR|USD|MR14</CreditCurveId>
  <IssuerId>CompanyXZY</IssuerId>
  <SettlesAccrual>true</SettlesAccrual>
  <FixedRecoveryRate>0.6</FixedRecoveryRate>
  <ProtectionFee>
    <LegData>
      <LegType>Cashflow</LegType>
      <Payer>true</Payer>
      <Currency>EUR</Currency>
      <CashflowData>
        <Cashflow>
          <Amount date="2018-10-03">91171.72</Amount>
        </Cashflow>
      </CashflowData>
    </LegData>
  </ProtectionFee>
  <Underlying>
    <!-- Alternatives:
         - Sequence of LegData, possibly with OptionData to represent callability
         - A single block of TreasuryLockData -->
    <OptionData> ... </OptionData>
    <NakedOption> ... </NakedOption>
    <LegData>
      <LegType>Floating</LegType>
      ...
    </LegData>
    <LegData>
      <LegType>Fixed</LegType>
      <Payer>false</Payer>
      ...
    </LegData>
  </Underlying>
</RiskParticipationAgreementData>
```

- ParticipationRate: The rate reflecting the participation amount relative to the swap volume.

  Allowable values: Any number between 0 and 1.

- ProtectionStart: The date on which the protection starts (inclusive).

  Allowable values: Any valid date, see See `Date` in Table 13.

- ProtectionEnd: The date on which the protection ends (exclusive).

  Allowable values. Any valid date greater than the protection start date.

- CreditCurveId: Typically the RED-code of the underlying swap reference entity defining the default curve used for pricing. Other identifiers may be used as well, provided they are supported in the market data configuration.

  Allowable values: Any valid credit curve identifier.

- IssuerId [Optional]: An identifier for the underlying swap reference entity. For informational purposes and not used for pricing. Defaults to an empty string.

  Allowable values: Any string.

- SettlesAccrual [Optional]: Whether or not the accrued coupon of the protection fee is due in the event of a default. This defaults to `true` if not provided. Only applies to coupon legs (i.e. not simple cashflows) within the protection fee block, otherwise it is ignored.

  Allowable values: `true` or `false`

- FixedRecoveryRate [Optional]: This node holds the fixed recovery rate if the RPA assumes a fixed recovery to calculate the settlement amount in case of a default event. If the field is omitted the recovery rate associated to the credit curve is used instead.

  Allowable values: Any number between 0 and 1.

- ProtectionFee: The fees that are paid (if protection is bought) or received (if protection is sold). The fees are given by one or more legs as described under 2.3.3 with identical Payer flags, typically this will be a single `Cashflow` leg holding zero or more fixed fee amounts or a `Fixed` leg representing a series of periodic fee payments. Fees are paid up to (but excluding) the default event. If the fees are given as coupons the accrued amount between the accrual start date and the default date is paid if and only if `SettlesAccrual` is set to `true`. The protection fees can be given in any arbitrary currency.

- Underlying: The reference underlying. There are several subtypes to distinguish, all of which have separate pricing engines attached. There is no need to specify the subtype in the trade xml, this is deduced automatically during the trade building:

  - Vanilla Swap: This is a vanilla swap given by two legs in the same currency, one receiver, one payer and one Fixed (or Cashflow), one Floating. For the floating part only Ibor coupons (no averaging) or (compounded, averaging) OIS coupons are allowed. Spreads and gearings are allowed, but no embedded caps/floors, no in arrears fixings for Ibor coupons. This type allows an analytic Black engine where the RPA Options are found via a representative swaption matching.

  - Structured Swap: As vanilla, but an arbitrary number of legs of type Fixed, Floating, Cashflow is allowed. Embedded caps/floors/collars and in arrears fixing are allowed. For floating legs, Ibor (no averaging) and OIS (compounded, averaging) coupons are allowed. All legs must be in the same currency. Standalone caps, floors, collars are allowed as an underlying of the RPA, if specified by a floating leg with NakedOption set to true. See 2.3.6 for details on the floating leg specification, amd likewise 2.3.5 for the fixed leg and 2.3.3 for the cashflow leg. This type requires a numeric grid engine.

  - Callable Swap / Swaption: As structured swap, but an additional OptionData block allows to specify callability of the swap. The relevant fields in OptionData are the same as for callable swaps, see 2.2.6. This type

requires a numeric grid engine as the structured swap. If NakedOption is set to true, an option to exercise into the underlying swap is represented, i.e. a swaption.

- Cross Currency Swap: Underlying legs as in structured swap, but the legs can be in two different currencies. No optionality is allowed though. At most two different currencies are allowed. This type can be priced using an analytic Black engine which models the FX Risk and assumes deterministic interest rates.
- T-Lock. The underlying is a T-Lock, represented as shown in listing 95 and explained in more detail below. This type requires a numeric grid engine.

Treasury Lock Underlying Specification

Listing 95 shows the specification of a T-Lock underlying. The fields have the following meaning:

- Payer: Boolean, true if the fixed reference rate is paid, false otherwise. I.e. if the payer flag is true and the yield is lower than the reference rate, then the underlying T-Lock trade pays the amount $(r - y) \cdot d$ where r is the reference rate, y is the yield, both expressed in basis points, and $d > 0$ is the (absolute) price change of the treasury bond when the yield moves by 1 basis point. Likewise, if the yield is higher than the reference rate, the underlying T-Lock trade receives $(y - r) \cdot d$.
  Allowable values: *true* or *false*
- BondData: Reference to the underlying security, given in in the BondData sub node, minimum required data are notional and security ID
  Allowable values: See 2.2.40
- ReferenceRate: Fixed rate paid or received on the T-Lock underlying
  Allowable values: Any real number. The rate is expressed in decimal form, eg 0.05 is a rate of 5%
- DayCounter [Optional]: Reference rate day counter. Optional, defaults to the coupon day counter of the underlying bond.
  Allowable values: See Table 18
- TerminationDate: Date for the cash settlement amount calculation
  Allowable values: See `Date` in Table 13.
- PaymentGap [Optional]: Business day gap between termination and payment date. Optional, defaults to zero.
  Allowable values: Any non-negative integer
- PaymentCalendar: Calendar to determine the payment date.
  Allowable values: See Table 17.

Listing 95: Treasury-Lock Data

```
<Underlying>
  <TreasuryLockData>
  <Payer>true</Payer>
  <BondData>
  ...
  </BondData>
  <ReferenceRate>0.05</ReferenceRate>
  <DayCounter>A360</DayCounter>
  <TerminationDate>2022-01-05</TerminationDate>
  <PaymentGap>5</PaymentGap>
  <PaymentCalendar>US</PaymentCalendar>
  </TreasuryLockData>
</Underlying>
```

2.2.57 Credit Linked Swap

Payoff

A credit linked swap is a swap where the payments are contingent on credit events occuring for a reference CDS. There are three classes of payments.

- Payments that are made only if no credit event has occured for a reference CDS until the payment date. If a credit event occurs within a coupon period the accrued amount until the credit event date might have to be paid or not, dependent on the terms of the trade. The accrual payment is done on the credit event date (with a small settlement delay in practice).
- Payments that are made in case a credit *has* occured, the amount is then weighted by either
  - the actual recovery rate RR associated to the credit event or
  - the loss given default rate $1 - RR$ based on the actual recovery rate or
  - a fixed (digital) recovery rate RR, agreed upon in the trade terms or
  - a fixed loss given default rate $1 - RR$

  These payments can be agreed to be made on the default date (with a small settlement delay in practice), on the period end date for the respective payment or on the trade maturity.
- Payments that are made independently of any credit events occuring for the reference CDS.

The underlying swap legs can pay any kind of coupon (fixed rate, Ibor, CMS, ...) or fixed amounts.

Input

A credit linked swap, trade type `CreditLinkedSwap`, is set up using a `CreditLinkedSwapData` block as shown in listing 96. The elements have the following meaning:

- CreditCurveId: The referenced CDS credit curve.

Allowable values: See `CreditCurveId` for credit trades - single name in Table 23. A `ReferenceInformation` node may be used in place of this `CreditCurveId` node.

- SettlesAccrual [Optional]: A flag indicating whether accrued coupon amounts are paid in case of a credit event. Optional, defaults to `true`. Applies to the payments specified under ContingentPayments.
  Allowable values: true, false
- FixedRecoveryRate [Optional]: A fixed (digital) recovery rate to apply. If not given, the market recovery rate is used. Applies to the payments specified under DefaultPayments and RecoveryPayments.
  Allowable values: Any non-negative real number.
- DefaultPaymentTime [Optional]: Controls the timing of the payments specified under DefaultPayments and RecoveryPayments. Defaults to `atDefault`.
  Allowable values: `atDefault`, `atPeriodEnd`, `atMaturity`.
- IndependentPayments [Optional]: The legs for which payments are made independent from credit events. The node contains one or more `LegData` subnodes representing these legs. Optional, can be omitted if no such payments are made.
  Allowable values: See 2.3.3 for the `LegData` subnode structure.
- ContingentPayments [Optional]: The legs for which payments are contingent on no credit event having occured until the payment date. If no such payments are made, the node can be omitted.
  Allowable values: See 2.3.3 for the `LegData` subnode structure.
- DefaultPayments [Optional]: The legs for which payments are contingent on a credit event having occured. If no such payments are made, the node can be omitted. If a default happens at a date d, the associated payment is the earliest payment with date greater or equal to d.
  Allowable values: See 2.3.3 for the `LegData` subnode structure.
- RecoveryPayments [Optiopnal]: The legs for which payments are contingent on a credit event having occured. The node works analogously to the DefaultPayments node, the only difference is that that payment amounts are weighted by RR instead of $1 - RR$.
  Allowable values: See 2.3.3 for the `LegData` subnode structure.

All legs must be given in the same currency.

Listing 96: Credit Linked Swap Data

```
<CreditLinkedSwapData>
  <CreditCurveId>RED:46A844|SNRFOR|USD|XR14</CreditCurveId>
  <SettlesAccrual>false</SettlesAccrual>
  <FixedRecoveryRate>0.4</FixedRecoveryRate>
  <DefaultPaymentTime>atDefault</DefaultPaymentTime>
  <IndependentPayments>
    <LegData> ... </LegData>
    <LegData> ... </LegData>
    ...
  </IndependentPayments>
  <ContingentPayments>
    <LegData> ... </LegData>
    <LegData> ... </LegData>
    ...
  </ContingentPayments>
  <DefaultPayments>
    <LegData> ... </LegData>
    <LegData> ... </LegData>
    ...
  </DefaultPayments>
  <RecoveryPayments>
    <LegData> ... </LegData>
    <LegData> ... </LegData>
    ...
  </RecoveryPayments>
</CreditLinkedSwapData>
```

2.2.58 Commodity Forward

A Commodity Forward contract is an agreement between two counterparties to buy/sell a set amount of a commodity, at a predetermined price (the strike), at the end of the contract. A commodity forward does not involve any upfront payment.

The `CommodityForwardData` node is the trade data container for the `CommodityForward` trade type. The structure of an example `CommodityForwardData` node is shown in Listings 97 and 98.

Listing 97: Commodity Forward data

```
<CommodityForwardData>
  <Position>Long</Position>
  <Maturity>2029-06-30</Maturity>
  <Name>XCEC:GC</Name>
  <Currency>USD</Currency>
  <Strike>1355</Strike>
  <Quantity>1000</Quantity>
  <IsFuturePrice>...</IsFuturePrice>
  <FutureExpiryDate>...</FutureExpiryDate>
  <FutureExpiryOffset>...</FutureExpiryOffset>
  <FutureExpiryOffsetCalendar>...</FutureExpiryOffsetCalendar>
  <PhysicallySettled>...</PhysicallySettled>
  <PaymentDate>...</PaymentDate>
</CommodityForwardData>
```

Listing 98: `CommodityForwardData` for forward on LME Aluminium 3M future.

```
<CommodityForwardData>
  <Position>Long</Position>
  <Maturity>2029-08-16</Maturity>
  <Name>XLME:AH</Name>
  <Currency>USD</Currency>
  <Strike>2160</Strike>
  <Quantity>1000</Quantity>
  <IsFuturePrice>true</IsFuturePrice>
  <FutureExpiryDate>2021-11-16</FutureExpiryDate>
  <PhysicallySettled>true</PhysicallySettled>
</CommodityForwardData>
```

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

- Position: Defines whether the underlying commodity will be bought (long) or sold (short).
  Allowable values: *Long, Short*
- Maturity: The maturity date of the forward contract, i.e. the date when the underlying commodity will be bought/sold.
  Allowable values: Any date string, see `Date` in Table 13.
- Name: The name of the underlying commodity.
  Allowable values: See `Name` for commodity trades in Table 25.
- Currency: The currency the underlying commodity is quoted in. The Strike and the Forward price (or Future price) of the underlying commodity are both considered to be in this currency.
  Allowable values: See `Currency` in Table 13.
- Strike: The agreed buy/sell price of the commodity forward.
  Allowable values: Any positive real number.
- Quantity: The number of units of the underlying commodity to be bought/sold.
  Allowable values: Any positive real number.
- `IsFuturePrice` [Optional]: This should be set to `true` if the forward contract underlying is the settlement price of a commodity future contract. If omitted, it defaults to `false`.
  Allowable values: Any string that evaluates to true or false as outlined in Table 29.
- `FutureExpiryDate` [Optional]: If `IsFuturePrice` is set to `true`, this gives the expiration date of the underlying commodity future contract. If omitted, the expiration date of the underlying commodity future contract is set equal to the value in the `Maturity` node. If `FutureExpiryDate` is provided, it takes precedence over any value provided in the `Maturity`, `FutureExpiryOffset` or `FutureExpiryOffsetCalendar` fields.
  Allowable values: Any date string, see `Date` in Table 13.
- `FutureExpiryOffset` [Optional]: If `IsFuturePrice` is set to `true` and

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

- FutureExpiryOffsetCalendar [Optional]: If FutureExpiryOffset is provided and is being used, this gives the calendar that should be used when generating the underlying commodity future contract expiration date from the Maturity date. If omitted, all days are considered good business days when generating the commodity future contract expiration date which is generally not what is desired.
  Allowable values: Any calendar string, see Calendar in Table 17.
- PhysicallySettled [Optional]: A value of true indicates that the forward contract is physically settled e.g. if the underlying is a future contract, that future contract is entered into on the Maturity date. A value of false indicates that the forward contract is cash settled e.g. if the underlying is a future contract, that future contract settlement price is observed on the Maturity date (or the FutureExpiryDate, when given) and the net amount due is exchanged on the cash settlement date. If omitted, it defaults to *true*.
  Allowable values: Any string that evaluates to true or false as outlined in Table 29.
- PaymentDate [Optional]: If PhysicallySettled is set to *false*, this gives the cash settlement date. It must be greater than or equal to the Maturity date. If omitted and the forward is cash settled, the Maturity date is used.
  Allowable values: Any date string, see Date in Table 13.
- SettlementData [Optional]: This node is used to specify the settlement of the cash flows for cash settled forwards, and the payment flow for physically settled ones.

A SettlementData node is shown in Listing 99, and the meanings and allowable values of its elements follow below.

- PayCurrency: The settlement currency for the payment cashflow.
  Allowable values: See Currency in Table 13.
- FXIndex: The FX reference index for determining the FX fixing at the value date. The Forward Price will be observed at maturity date (or future expiry date if it's a future), the NPV is converted to PayCurrency with the FXIndex using an FX fixing on FixingDate (settlement date) discounted from PaymentDate.
  Allowable values: The format of the FXIndex is "FX-FixingSource-CCY1-CCY2" as described in Table 21.
- FixingDate: The date on which the *FXIndex* is observed. Allowable values: See Date in Table 13.

Listing 99: Example `SettlementData` node with `Rules` sub-node

```
<SettlementData>
  <PayCurrency>EUR</PayCurrency>
  <FXIndex>FX-ECB-EUR-USD</FXIndex>
  <FixingDate>2025-05-28</FixingDate>
</SettlementData>
```

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

2.2.59 Commodity Swap and Basis Swap

Payoff

A Commodity Swap involves the exchange of floating commodity prices against a fixed known commodity price, with cash settlement either at the end of the swap or on a monthly basis. A long position in a commodity swap involves computing the arithmetic average of the difference between the variable fixing and the fixed strike K at each of the fixing dates t_i $(i = 1, ..., n)$

$$V_i = S_{t_i} - K$$

in the calculation period. S_{t_i} denotes the relevant fixing on price date t_i, which may be a commodity spot price (if available), possibly the average of high and low values on that day in a given source. Or the relevant fixing S_{t_i} may be the price of the *prompt future* $f_{t_i,T(t_i)}$ where $T(t_i)$ is the earliest futures expiry after t_i.

The final payoff at calculation period end will be known at period end and given by the arithmetic average

$$V = \frac{1}{n}\sum_{i=1}^{n} V_i.$$

This amount is settled in cash with a delay after the end of the calculation period or rolled up into a single payment after swap end.

A Commodity **Basis Swap** involves the exchange of the spread between floating commodity prices against a fixed known spread K, with cash settlement either at the end of the swap or on a monthly basis. Generalizing the Commodity Swap above, the payoff contribution on each pricing date is now

$$V_i = S^{(1)}_{t_i} - S^{(2)}_{t_i} - K$$

where $S^{(1,2)}$ may be spot or prompt futures prices. The final payoff at period end will be known at period end and given by the arithmetic average

$$V = \frac{1}{n}\sum_{i=1}^{n} V_i.$$

Input

The structure of a CommoditySwap trade node is shown in listing 100. This trade node can be used to represent commodity swaps and commodity basis swaps. It consists of the generic Envelope and the specific SwapData section.

The SwapData node may contain two or more LegData nodes. There must be at least one LegData node of a commodity LegType, i.e. CommodityFixed or CommodityFloating, but non-commodity leg types are also allowed. The commodity leg types are described in sections 2.3.20 and 2.3.22 respectively.

Listing 100: Commodity Swap

```
<Trade id="...">
  <TradeType>CommoditySwap</TradeType>
  <Envelope>
  </Envelope>
  <SwapData>
    <LegData>
      <LegType>CommodityFixed</LegType>
      ...
    </LegData>
    <LegData>
      <LegType>CommodityFloating</LegType>
      ...
    </LegData>
  </SwapData>
</Trade>
```

2.2.60 Commodity Swaption

Payoff

A Commodity Swaption is a European option on a forward starting Commodity Swap 2.2.59 consisting of a sequence of calculation periods. The exercise decision is made once before the Swap starts. In a Payer Swaption, the holder of the option has the right to pay the fixed prices and to receive the floating prices. In a Receiver Swaption, the holder of the option has the right to receive the fixed prices and to pay the floating prices.

Input

The structure of a trade node representing a commodity swaption is shown in listing 101. It consists of the generic Envelope and the specific CommoditySwaptionData node.

The CommoditySwaptionData node contains an OptionData node described in 2.3.1. The relevant fields in the OptionData node for a CommoditySwaption are:

- LongShort: The allowable values are *Long* or *Short*. Note that the payer and receiver legs in the underlying swap are always from the perspective of the party that is *Long*. E.g. for a *Short* CommoditySwaption with a fixed leg where the Payer flag is set to *false*, it means that the counterparty receives the fixed flows.
- OptionType[Optional]: This flag is optional for CommoditySwaptions, and even if set, has no impact. The direction of flows is determined entirely by the Payer flags on the underlying legs (and the LongShort flag above).

- `Style`: The exercise style of the CommoditySwaption. Only exercise style *European* is supported.
- `NoticePeriod`[Optional]: The notice period defining the date (relative to the exercise date) on which the exercise decision has to be taken. If not given the notice period defaults to *0D*, i.e. the notice date is identical to the exercise date. Allowable values: A number followed by *D, W, M, or Y*
- `NoticeCalendar`[Optional]: The calendar used to compute the notice date from the exercise date. If not given defaults to the *NullCalendar* (no holidays, weekends are no holidays either). Allowable values: See Table 17 `Calendar`.
- `NoticeConvention`[Optional]: The roll convention used to compute the notice date from the exercise date. Defaults to *Unadjusted* if not given. Allowable values: See Table 14 Roll Convention.
- `Settlement`: Delivery Type. The allowable values are *Cash* or *Physical*.
- `ExerciseFees`[Optional]: This node contains child elements of type `ExerciseFee`. Similar to a list of notionals (see 2.3.3) the fees can be given either
  - as a list where each entry corresponds to an exercise date and the last entry is used for all remaining exercise dates if there are more exercise dates than exercise fee entries, or
  - using the `startDate` attribute to specify a change in a fee from a certain day on (w.r.t. the exercise date schedule)

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

  If a fee is given as a positive number the option holder has to pay a corresponding amount if they exercise the option. If the fee is negative on the other hand, the option holder receives an amount on the option exercise.
- `ExerciseFeeSettlementPeriod`[Optional]: The settlement lag for exercise fee payments. Defaults to 0D if not given. This lag is relative to the exercise date (as opposed to the notice date). Allowable values: A number followed by *D, W, M, or Y*
- `ExerciseFeeSettlementCalendar`[Optional]: The calendar used to compute the exercise fee settlement date from the exercise date. If not given defaults to the *NullCalendar* (no holidays, weekends are no holidays either). Allowable values: See Table 17 Calendar.
- `ExerciseFeeSettlementConvention`[Optional]: The roll convention used to compute the exercise fee settlement date from the exercise date. Defaults to *Unadjusted* if not given. Allowable values: See Table 14 Roll Convention.
- An `ExerciseDates` node where exactly one `ExerciseDate` date element must be given for *European* style CommoditySwaptions. Allowable values: The

`ExerciseDate` must be on or before the StartDate of the underlying legs, and be on or after the valuation date. For the format, see Date in Table 13.

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

  Allowable values: See section 2.3.2

The `CommoditySwaptionData` node should contain exactly two `LegData` nodes. One `LegData` node should be of type `CommodityFixed` described in section 2.3.20 and one should be of type `CommodityFloating` described in section 2.3.22. Note that on the `CommodityFloating` leg, the Spread must be omitted or set to *0*, and the Gearing must be omitted or set to *1*.

Listing 101: Commodity swaption

```
<Trade id="...">
  <TradeType>CommoditySwaption</TradeType>
  <Envelope>
    ...
  </Envelope>
  <CommoditySwaptionData>
    <OptionData>
      <LongShort>Long</LongShort>
      <Style>European</Style>
      <Settlement>Cash</Settlement>
      <ExerciseDates>
        <ExerciseDate>2023-01-05</ExerciseDate>
      </ExerciseDates>
    </OptionData>
    <LegData>
      <LegType>CommodityFixed</LegType>
      ...
    </LegData>
    <LegData>
      <LegType>CommodityFloating</LegType>
      ...
    </LegData>
  </CommoditySwaptionData>
</Trade>
```

2.2.61 Commodity Option

A European Commodity Option gives the buyer the right, but not the obligation, to buy a set amount of a commodity, at a predetermined price (the strike), at the end of the contract. For this right the buyer pays a premium to the seller. Settlement can be either cash or physical delivery.

The `CommodityOptionData` node is the trade data container for the *CommodityOption* trade type. Vanilla commodity options are supported. The exercise style may be *European* or *American*. The `CommodityOptionData` node includes exactly one `OptionData` trade component sub-node plus elements specific to the commodity option. The structure of a `CommodityOptionData` node for a commodity option is shown in Listing 102.

Listing 102: Commodity Option data

```
<CommodityOptionData>
  <OptionData>
   <LongShort>Short</LongShort>
   <OptionType>Put</OptionType>
   <Style>European</Style>
   <Settlement>Cash</Settlement>
   <PayOffAtExpiry>false</PayOffAtExpiry>
    <ExerciseDates>
      <ExerciseDate>2029-04-28</ExerciseDate>
     </ExerciseDates>
  </OptionData>
  <Name>NYMEX:CL</Name>
  <Currency>USD</Currency>
  <StrikeData>
    <StrikePrice>
      <Value>100</Value>
      <Currency>USD</Currency>
    </StrikePrice>
  </StrikeData>
  <Quantity>500000</Quantity>
  <IsFuturePrice>true<IsFuturePrice>
  <FutureExpiryDate>2029-04-28<FutureExpiryDate>
</CommodityOptionData>
```

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

- The `CommodityOptionData` node contains an `OptionData` node described in 2.3.1. The relevant fields in the `OptionData` node for a CommodityOption are:
  - `LongShort`: The allowable values are *Long* or *Short*.
  - `OptionType`: The allowable values are *Call* or *Put*.
  - `Style`: The exercise style of the CommodityOption. The allowable values are *European* or *American*.
  - `PayOffAtExpiry`: This must be set to *false* as payoff at expiry is not currently supported.
  - An `ExerciseDates` node where exactly one `ExerciseDate` date element must be given for. Allowable values: See Date in Table 13.
  - `Premiums` [Optional]: Option premium node with amounts paid by the option buyer to the option seller. Allowable values: See section 2.3.2
- Name: The name of the underlying commodity.
  Allowable values: See `Name` for commodity trades in Table 25.
- Currency: The currency of the commodity option.
  Allowable values: See `Currency` in Table 13.
- StrikeData: The option strike price. It uses the price quotation outlined in the underlying contract specs for the commodity name in question.
  Allowable values: Only supports `StrikePrice` as described in Section 2.3.30.

- Quantity: The number of units of the underlying commodity covered by the transaction. The unit type is defined in the underlying contract specs for the commodity name in question. For avoidance of doubt, the Quantity is the number of units of the underlying commodity, not the number of contracts. Allowable values: Any positive real number.
- IsFuturePrice [Optional]: A boolean indicating if the underlying is a future contract settlement price, `true`, or a spot price, `false`.

  Allowable values: A boolean value given in Table 29. If not provided, the default value is `true`.
- FutureExpiryDate [Optional]: If `IsFuturePrice` is `true` and the underlying is a future contract settlement price, this node allows the user to specify the expiry date of the underlying future contract.

  Allowable values: This should be a valid date as outlined in Table 13. If not provided, it is assumed that the future contract's expiry date is equal to the option expiry date provided in the `OptionData` node.

2.2.62 Commodity Digital Option

A commodity digital option is represented with trade type *CommodityDigitalOption* and a corresponding `CommodityDigitalOptionData` node. The latter differs from the `CommodityOptionData` node in section 2.2.61 by replacing tag *Quantity* with tag *Payoff* which is the cash amount paid in the Currency of the option from the party that is short to the party that is long, when the underlying price exceeds the strike at expiry in case of a Call (or falls below the strike in case of a Put). The digital option is priced in ORE as a spread of vanilla Commodity options at two slightly different strikes. For option type *Call* and *Put*, respectively, the digital call/put is constructed as

$$\text{Digital Call} = \frac{\text{Payoff}}{\Delta} \times (\text{Call}(K - \Delta/2) - \text{Call}(K + \Delta/2))$$
$$\text{Digital Put} = \frac{\text{Payoff}}{\Delta} \times (\text{Put}(K + \Delta/2) - \text{Put}(K - \Delta/2))$$

so that the long digital option has positive value in both cases. The strike spread Δ used here is set to 1% of strike K. However, the default 1% can be overriden by setting the 'StrikeSpread' parameter in the pricing engine global parameters.

2.2.63 Commodity Spread Option

A commodity Spread Option is represented with trade type *CommoditySpreadOption* and a corresponding `CommoditySpreadOptionData` node.

The `CommoditySpreadOptionData` node is the trade data container for the *CommoditySpreadOption* trade type. The structure of a `CommoditySpreadOptionData` node for a commodity option is shown in Listing 103.

The `CommoditySpreadOptionData` include exactly two `LegData` nodes of type *CommodityFloating*. Details on these are described in 2.3.22. The resulting Legs must produce the same amount of cashflows (i.e. the number of *calculation period*s must be

the same for the long and short positions). If the number of cashflows per leg is 1, this trade represents a single commodity spread option. If is greater than 1, it represents a multi-period commodity spread option. Exactly one payer and one receiver leg are required, the leg with `isPayer` setto *true* is the long (positive) position in the spread payoff.

Within the two `LegData`, the `Quantity` node has must be equal. If the underlying contracts are quoted using different units (e.g. barrels vs liters), the `Gearing` node must be used to account for this difference. The gearing could also be used for the heat rate factor in spark / heat rate options.

Other than the two legs, the following nodes complete the `CommoditySpreadOptionData` container:

- `SpreadStrike`: The strike value for the spread. Allowable values: Any real number.
- `OptionData`: This is a trade component sub-node outlined in section 2.3.1. The relevant fields in the `OptionData` node for an CommoditySpreadOption are
  - `LongShort` The allowable values are *Long* or *Short*.
  - `OptionType` The allowable values are *Call* or *Put*.
  - `Premiums` [Optional]: Option premium amounts paid by the option buyer to the option seller. See section 2.3.2
  - `ExerciseDates` [Optional]: If given, there must be one exercise date for each option period. If omitted, the option expiry is derived from the cashflows and falls on the last pricing period of the commodity cashflow.
  - `PaymentData` [Optional]: This node is used to supply the date on which the option is cash settled if it is exercised. If omitted the settlement date is derived from the cashflow payment dates.
- `OptionStripPaymentDates` [Optional]: If the number of cashflows per leg is greater than 1, we can group options by their expiry date into strips. All option in a strip will have the same payment date as defined in this node. The payment date will be *lag* business days after the latest expiry date in the strip. The node has following sub-nodes:
  - `OptionStripDefinition` A schedule node 2.3.4 defining the option strips. The n dates in the schedule defining $n-1$ strips, each strip include the period's start date and excludes period's end date. All options with expiry within start and end of a period are falling in the same strip. The schedule has to cover all option expiries. The first date in the schedule has to be before or on the first expiry date of the options and the last date in the schedule has to be after last expiry date of the options.
  - `PaymentCalendar` Calendar defining valid business days for the payment date.
  - `PaymentLag` number of business days.

– `PaymentConvention` business day convention for the option strip payment date.

Listing 103: Commodity Spread Option data

```
<CommoditySpreadOptionData>
  <LegData>
   <LegType>CommodityFloating</LegType>
   <IsPayer>true<IsPayer>
   ...
  </LegData>
  <LegData>
   <LegType>CommodityFloating</LegType>
   <IsPayer>false<IsPayer>
   ...
  </LegData>
  <OptionData>
    <LongShort>Long</LongShort>
    <OptionType>Call</OptionType>
     <Premiums>
       <Premium>
         <Amount>10900</Amount>
         <Currency>EUR</Currency>
         <PayDate>2020-03-01</PayDate>
       </Premium>
     </Premiums>
  </OptionData>
  <SpreadStrike>2.3</SpreadStrike>
  <OptionStripPaymentDates>
          <OptionStripDefinition>
          <Rules>
            <StartDate>2023-07-01</StartDate>
            <EndDate>2023-10-01</EndDate>
            <Tenor>1M</Tenor>
            <Calendar>NullCalendar</Calendar>
            <Convention>Unadjusted</Convention>
            <TermConvention>Unadjusted</TermConvention>
            <Rule>Backward</Rule>
          </Rules>
          </OptionStripDefinition>
    <PaymentCalendar>ICE_FuturesUS,US-NERC</PaymentCalendar>
    <PaymentLag>5</PaymentLag>
    <PaymentConvention>MF</PaymentConvention>
  </OptionStripPaymentDates>
</CommoditySpreadOptionData>
```

2.2.64 Commodity Average Price Option

Payoff

The (Arithmetic) Average Price Option (APO) is an Asian option with a single-period Commodity Swap as an underlying which observes daily price fixings in the calculation period. Option exercise is at calculation period end, typically on the last floating price fixing date - in contrast to a Commodity Swaption 2.2.60 where expiry is before underlying start. In a call option, the owner of the option has the right to receive average floating prices and to pay fixed prices. In a put option it is the other way

around.

APOs can occur in combinations, e.g. series of put and call options for different strike prices, and for a series of consecutive calculation periods.

Input

The structure of a trade node representing a commodity average price option (APO) is shown in listing 104. It consists of the generic `Envelope` and the specific `CommodityAveragePriceOptionData` node. A strip of these options may be booked using the commodity option strip trade outlined in section 2.2.65. The meanings and allowable values of the elements in the `CommodityAveragePriceOptionData` are as follows:

- `OptionData`: This node is described in section 2.3.1. The relevant fields in the `OptionData` node for a CommodityAveragePriceOption are:
  - `LongShort`: The allowable values are *Long* or *Short*.
  - `OptionType`: The allowable values are *Call* or *Put*, where *Call* is an option for the party that is *Long* to buy the underlying commodity, and *Put* is an option to sell the underlying commodity.
  - `Style`: only `European` exercise style is supported.
  - the `ExerciseDates` node should contain exactly one `ExerciseDate`, and the single `ExerciseDate` should be on or before the payment date. Also, the single `ExerciseDate` should be on or after the final date in the calculation period i.e. the `EndDate` below.
  - `Premiums` [Optional]: Option premium node with amounts paid by the option buyer to the option seller. Allowable values: See section 2.3.2
- `BarrierData` [optional]: If given this node specifies the barrier terms of the option:
  - `Type`: One of *UpAndIn*, *DownAndIn*, *UpAndOut*, *DownAndOut*
  - `Style`: One of *European*, *American*. A European barrier is observed on the last relevant pricing date of the APO while an American barrier is observed on all pricing dates of the APO.
  - `LevelData`: The barrier level. Only single-barrier options are allowed, i.e. exactly one level must be given.
- `Name`: An identifier specifying the commodity being referenced. This is described in section 2.3.24.

  Allowable values: See `Name` for commodity trades in Table 25.
- `Currency`: The currency of the payoff which must be consistent with either currency of the market data set up for the commodity or the other currency specified in `FXIndex` (see below).

  Allowable values: See `Currency` in Table 13.

- `Quantity`: The number of units of the underlying commodity covered by the APO. The unit type is defined in the underlying contract specs for the commodity name in question. For avoidance of doubt, the Quantity is the number of units of the underlying commodity, not the number of contracts.

  The meaning of the Quantity is influenced by the `CommodityQuantityFrequency` value as described in section 2.3.24. If `CommodityQuantityFrequency` is set to `PerCalculationPeriod`, this quantity is used directly in the APO payoff. If `CommodityQuantityFrequency` is set to `PerCalendarDay`, this quantity is multiplied by the number of calendar days in the APO period to give the final quantity that is used in the APO payoff.

  Allowable values: Any positive real number.

- `StrikeData`: A `StrikeData` node is used as described in Section 2.3.30 to represent the APO strike price and Currency of the Strike.The APO strike price. The strike uses the price quotation outlined in the underlying contract specs for the commodity name in question. Note that for CommodityAPOs only `StrikePrice` is supported within the `StrikeData` node, and not `StrikeYield`.

  Allowable values: Only supports `StrikePrice` as described in Section 2.3.30.

- `PriceType`: The price type is `Spot` if the APO is referencing a commodity spot price, and it is `FutureSettlement` if the APO is referencing a commodity future contract settlement price.

  Allowable values: `Spot` or `FutureSettlement`.

- `StartDate`: The start date of the APO's calculation period.

  Allowable values: See `Date` in Table 13.

- `EndDate`: The end date of the APO's calculation period.

  Allowable values: See `Date` in Table 13.

- `PaymentCalendar`: The business calendar used to determine the valid payment date.

  Allowable values: See Table 17 `Calendar`.

- `PaymentLag`: The payment date is this number of business days after a given base date. The base date is determined by the value of the `CommodityPayRelativeTo` node below which is generally omitted for APOs and allowed to take its default value of `CalculationPeriodEndDate`.

  Allowable values: Any valid period, i.e. a non-negative whole number, optionally followed by D (days), W (weeks), M (months), Y (years). Defaults to *0D* if left blank or omitted. If a whole number is given and no letter, it is assumed that it is a number of D (days).

- `PaymentConvention`: The roll convention used to adjust the payment date.

  Allowable values: See Table 14 Roll Convention. Defaults to *Unadjusted* if left blank or omitted.

- `PricingCalendar`: The business calendar used for determining the *Pricing Dates* in the calculation period.

  Allowable values: See Table 17 `Calendar`. Defaults to *NullCalendar* (no holidays) if left blank or omitted.

- `PaymentDate` [Optional]: An explicit payment date for the APO if the combination of `PaymentCalendar`, `PaymentLag` and `PaymentConvention` is not sufficient. If `PaymentDate` is provided, it overrides the values provided in `PaymentCalendar`, `PaymentLag` and `PaymentConvention`.

- `Gearing` [Optional]: An optional gearing factor that the average price is multiplied by in the APO payoff. The default value is 1.0.

- `Spread` [Optional]: An optional spread that is added to the average price in the APO payoff. The default value is 0.0.

- `CommodityQuantityFrequency` [Optional]: This is as described in section 2.3.24.

- `CommodityPayRelativeTo` [Optional]: This is as described in section 2.3.24.

- `FutureMonthOffset` [Optional]: This is as described in section 2.3.24. Note that IsAveraged defaults to *false* as it cannot be used as a tag within the CommodityAveragePriceOptionData node. Thus, if e.g. FutureMonthOffset is set to *2*, the future contract month and year is taken as the second month following the base date's month and year; and so on for all positive values of FutureMonthOffset.

- `DeliveryRollDays` [Optional]: This is as described in section 2.3.24.

- `IncludePeriodEnd` [Optional]: This is as described in section 2.3.24.

- `FXIndex` [Optional]: This is an FX index used to apply currency conversion daily in the average. The currencies pair must include the currency used in the underlying commodity trade and the currency used for settlement.

  Allowable values: See Table 21 for supported fx indices.

2.2.65 Commodity Option Strip

The structure of a trade node representing a commodity option strip is shown in listing 105. This node can be used to represent a strip of commodity average price options as described in section 2.2.64 or a strip of European commodity options as described in section 2.2.61. It consists of the generic `Envelope` and the specific `CommodityOptionStripData` node.

The `CommodityOptionStripData` node has a `LegData` node with `LegType` set to `CommodityFloating`. This `LegData` node is described in detail in sections 2.3.22 and 2.3.24. Note that the `Payer` field in `CommodityFloatingLegData`, while mandatory, has no impact on flows. The node `IsAveraged` in `CommodityFloatingLegData` determines whether a strip of European commodity options or a strip of APOs are created:

- If `IsAveraged` is `false`, a strip of European commodity options is created. There is a European put and or European call option created for each calculation period. The exercise date of the option in the calculation period is given by the

Listing 104: Commodity average price option

```
<Trade id="...">
  <TradeType>CommodityAveragePriceOption</TradeType>
  <Envelope>
  ...
  </Envelope>
  <CommodityAveragePriceOptionData>
    <OptionData>
      <LongShort>Short</LongShort>
      <OptionType>Call</OptionType>
      <Style>European</Style>
      <ExerciseDates>
         <ExerciseDate>2020-01-31</ExerciseDate>
      </ExerciseDates>
    </OptionData>
    <BarrierData>
      <Type>UpAndIn</Type>
      <Style>American</Style>
      <LevelData>
        <Level>
          <Value>80</Value>
        </Level>
      </LevelData>
    </BarrierData>
    <Name>NYMEX:CL</Name>
    <Currency>USD</Currency>
    <Quantity>6000</Quantity>
    <StrikeData>
      <StrikePrice>
        <Value>80</Value>
        <Currency>USD</Currency>
      </StrikePrice>
    </StrikeData>
    <PriceType>FutureSettlement</PriceType>
    <StartDate>2022-01-01</StartDate>
    <EndDate>2023-01-31</EndDate>
    <PaymentCalendar>USD</PaymentCalendar>
    <PaymentLag>5</PaymentLag>
    <PaymentConvention>Following</PaymentConvention>
    <PricingCalendar>USD</PricingCalendar>
    <Gearing>1</Gearing>
    <Spread>0.0</Spread>
    <CommodityQuantityFrequency>PerCalculationPeriod</CommodityQuantityFrequency>
    <CommodityPayRelativeTo>CalculationPeriodEndDate</CommodityPayRelativeTo>
    <FutureMonthOffset>0</FutureMonthOffset>
    <DeliveryRollDays>0</DeliveryRollDays>
    <IncludePeriodEnd>true</IncludePeriodEnd>
    <FXIndex>FX-ECB-EUR-USD</FXIndex>
  </CommodityAveragePriceOptionData>
</Trade>
```

Pricing Date in the calculation period using the rules outlined in section 2.3.24. The quantity is given by the quantity in the calculation period using the rules outlined in section 2.3.24. If cash settled, the cash settlement date is given by the payment date for the calculation period using the rules outlined in section 2.3.24.

- If `IsAveraged` is `true`, a strip of commodity average price options is created. There is a put and or call option created for each calculation period. The exercise date of the option in the calculation period is given by the calculation period end date. The quantity is given by the quantity in the calculation period using the rules outlined in section 2.3.24.

Each calculation period may contain a put and a call that may be either bought or sold. The type of option, whether they are bought or sold and the strike price is determined by the `Calls` and `Puts` nodes. We describe here the settings for the `Calls` node with the understanding that analogous descriptions apply to the `Puts` node. If the `Calls` node is omitted, it is assumed that there are no call options in the strip.

The `LongShorts` node may contain one `LongShort` node or the same number of `LongShort` nodes as calculation periods. Each `LongShort` node has the allowable values `Long` or `Short`. If `LongShort` is `Long`, then the call option is bought and if `LongShort` is `Short` then the call option is sold. If a single `LongShort` node is provided, it is applied to all options in the strip. If the same number of `LongShort` nodes as calculation periods are provided, a `LongShort` node is applied to the option in the corresponding period. The optional `BarrierData` node specifies the barrier terms of the options. See section 2.2.64 for details on this. Call and put options can have different barrier terms, but all call (resp. put) options share the same terms. In listing 105 only the call options have a barrier feature.

Similar to the `LongShorts` node, the `Strikes` node may contain one `Strike` node or the same number of `Strike` nodes as calculation periods. If only one is provided, this strike applies to all options in the strip. If the same number of `Strike` nodes as calculation periods are provided, a `Strike` node is applied to the option in the corresponding period. In this way, we support varying strikes across options in the strip. At least one of `Calls` or `Puts` needs to be provided for a valid option strip to be created.

The `Premiums` node allows for the addition of premiums. If the `PremiumAmount` is negative, it is paid and if it is positive, it is received. See 2.3.2.

The optional `Style` node can be set to `European` or `American` to change the exercise style for the strip of options. If not set, `European` is assumed. If the strip is a strip of APOs, `European` is assumed and a warning is issued if `Style` is not `European`.

The optional `Settlement` node can be set to `Cash` or `Physical` to change the settlement method for the strip of options. If not set, `Cash` is assumed. If the strip is a strip of APOs, `Cash` is assumed and a warning is issued if `Settlement` is not `Cash`.

The optional `IsDigital` node allows the creation of a strip of `CommodityDigitalOptions` (see 2.2.62). If set to `true` the node `PayoffPerUnit` needs to be set.

Node `PayoffPerUnit` [Optional] specifies the payoff per commodity unit, expressed in leg currency, in case a digital option is exercised. If the trade is a strip of digital options, this node must be set. It accepts real numbers as input.

Listing 105: Commodity option strip

```
<Trade id="...">
  <TradeType>CommodityOptionStrip</TradeType>
  <Envelope>
    ...
  </Envelope>
  <CommodityOptionStripData>
    <LegData>
      <LegType>CommodityFloating</LegType>
      ...
    </LegData>
    <Calls>
      <LongShorts>
        <LongShort>Short</LongShort>
      </LongShorts>
      <Strikes>
        <Strike>5.3</Strike>
      </Strikes>
      <BarrierData>
        <Type>UpAndIn</Type>
        <Style>American</Style>
        <LevelData>
          <Level>
            <Value>70.0</Value>
          </Level>
        </LevelData>
      </BarrierData>
    </Calls>
    <Puts>
      <LongShorts>
        <LongShort>Long</LongShort>
      </LongShorts>
      <Strikes>
        <Strike>8.17</Strike>
      </Strikes>
    </Puts>
    <Premiums> ... </Premiums>
    <Style>European</Style>
    <Settlement>Cash</Settlement>
  </CommodityOptionStripData>
</Trade>
```

2.2.66 Commodity Variance and Volatility Swap

A Commodity Variance or Volatility Swap has a payoff that depends on the volatility/variance of an underlying commodity instrument. See section 2.2.34 for the equivalent Equity product.

The `CommodityVarianceSwapData` node is the trade data container for the *CommodityVarianceSwap* trade type. The structure of an example `CommodityVarianceSwapData` node for a Commodity Variance Swap is the same as for an Equity Variance Swap in section 2.2.34, with the exception of the underlying node which is of type 'Commodity' here. See section 2.3.29 for additional optional elements of the underlying node and allowable values.

2.2.67 Commodity Position

An commodity position represents a position in a single commodity - using a single `Underlying` node, or in a weighted basket of underlying commodities - using multiple `Underlying` nodes.

An commodity Position can be used both as a stand alone trade type (TradeType: *CommodityPosition*) or as a trade component (`CommodityPositionData`) used within the *TotalReturnSwap* (Generic TRS) trade type, to set up for example Commodity Basket trades.

If the *PriceType* is set to *FutureSettlement* it will refer by default to today's prompt (lead) future. At the moment a generic TRS doesn't support rolling of the future contracts. Today's prompt future could be different from the prompt future at inception. If the initial price for the basket is not set, it will use the price of today's prompt future at trade inception as initial price and the TRS will also ignore the roll yield caused by rolling from one prompt future to the next contract.

It is set up using an `CommodityPositionData` block as shown in listing 106. The meanings and allowable values of the elements in the block are as follows:

- Quantity: The number of shares or units of the weighted basket held.
  Allowable values: Any positive real number
- Underlying: One or more underlying descriptions. If a basket of commodities is defined, the `Weight` field should be populated for each underlyings. The weighted basket price is then given by

$$\text{Basket-Price} = \text{Quantity} \times \sum_i \text{Weight}_i \times S_i \times \text{FX}_i$$

  where

  - S_i is the i-th commodity prompt future or spot price in the basket
  - FX_i is the FX Spot converting from the ith commodity currency to the first commodity currency which is by definition the currency in which the npv of the basket is expressed.

  Allowable values: See 2.3.29 for the definition of an underlying. Only commodity underlyings are allowed.

Listing 106: Commodity position data

```
<Trade id="CommodityPosition">
  <TradeType>CommodityPosition</TradeType>
  <Envelope>...</Envelope>
  <CommodityPositionData>
    <Quantity>1000</Quantity>
    <Underlying>
      <Type>Commodity</Type>
      <Name>NYMEX:CL</Name>
      <Weight>0.5</Weight>
      <PriceType>FutureSettlement</PriceType>
      <FutureMonthOffset>0</FutureMonthOffset>
      <DeliveryRollDays>0</DeliveryRollDays>
      <DeliveryRollCalendar>TARGET</DeliveryRollCalendar>
    </Underlying>
    <Underlying>
      <Type>Commodity</Type>
      <Name>ICE:B</Name>
      <Weight>0.5</Weight>
      <PriceType>FutureSettlement</PriceType>
      <FutureMonthOffset>0</FutureMonthOffset>
      <DeliveryRollDays>0</DeliveryRollDays>
      <DeliveryRollCalendar>TARGET</DeliveryRollCalendar>
    </Underlying>
  </CommodityPositionData>
</Trade>
```

2.2.68 Generic Total Return Swap / Contract for Difference (CFD)

A generic total return swap / CFD (Trade type: *TotalReturnSwap* or *ContractForDifference*) is set up using a TotalReturnSwapData (or ContractForDifferenceData) block as shown in listing 109 and 115. Both trade types behave exactly the same.

Usually CFDs are traded without a funding component and captured with only two dates in the return schedule, namely the start date on which the initial price is fixed and a fictitious closing date usually set to "tomorrow" or another suitable future date. See listing 115 for the setup of a CFD on STOXX50E with initial price 3399.20 on 2019-09-28.

The generic total return swap is priced using the *accrual method* as opposed to a *full discounting method* as it is used for the *equity swap* trade type. The accrual method is common practice when daily unwind rights are present in the trade terms or when the underlying valuation is too complex to allow for future projection.

The TotalReturnSwapData (ContractForDifferenceData) block is comprised of four sub-blocks, which are

- `UnderlyingData` containing one or more `Trade` subnodes describing the asset position of the TRS
- `ReturnData` describing the fixing and payment schedule of the return leg and specifying indices for FX conversion if applicable

- `FundingData` (optional) containing one or more funding legs of the TRS, whose notionals are based on either
  - "PeriodReset": the underlying price on the last valuation date before or on the accrual start date of the relevant funding coupon, this price is converted to the funding currency using the FX rate on this same valuation date for compo / cross currency swaps (see below)
  - "DailyReset": the underlying price on each day of the accrual period, again converted to the funding currency using the FX rate of the same date for compo / cross currency swaps. This notional type is only supported for fixed rate funding legs.
  - "Fixed": a fixed notional given explicitly in the funding leg
- `AdditionalCashflowData` (optional) a single leg of type Cashflow containing additional payments

The `ReturnData` and `FundingData` schedule periods often match, but this is not a strict requirement: In general, the funding notional is determined as described above dependent on the notional types "PeriodReset", "DailyReset", "Fixed".

Notice that in every case, the `UnderlyingData` schedule (if applicable to the underlying trade type as e.g. for a bond) is completely independent from the funding / return schedules: The underlying schedule defines the underlying flows to compute its NPV, and is not directly related to the return swap itself.

Generic TRS can be used to represent total return swaps on a wide range of underlying assets including e.g. single bonds or equities, CFDs on an underlying basket of EquityPositions, proprietary indices on equity options and equity or bond indices.

- The `UnderlyingData` block specifies one or more underlyings, which can be a trades of one of the following types (see the trade type specific sections), or structures with the Derivative or PortfolioIndexTradeData subnode.
  - Bond: See 2.2.40, the trade data is given in a BondData sub node for a single Bond.
  - ForwardBond: See 2.2.43, the trade data is given in a ForwardBondData sub node. To represent a TRS on a Bond Futures position a ForwardBond underlying is used.
  - CBO: See 2.2.50, the trade data is given in a CBOData sub node.
  - CommodityPosition: See 2.2.67, the trade data is given in a CommodityPositionData sub node.
  - ConvertibleBond: See 2.2.48, the trade data is given in a ConvertibleBondData sub node. When using reference data, a TRS on a convertible bond can also be captured as a TRS on a bond, i.e. there is no need to distinguish between a TRS on a Bond and a TRS on a convertible Bond in this case, the pricer will figure out which underlying to set up based on the type of reference data that is set up for the ISIN referenced in the security id field.

- EquityPosition: See 2.2.36, the trade data is given in a EquityPositionData sub node. Notice that the equities given in the basket must be available as quoted market data. To represent a TRS on an Equity Futures position, one or multiple EquityPosition underlyings are used, where the equity positions cover the underlying equity of the futures position.
- EquityOptionPosition: See 2.2.37, the trade data is given in a EquityOptionPositionData sub node.
- BondPosition: See 2.2.41, the trade data is given in a BondBasketData sub node for multiple underlying Bonds.
- CashPosition: See 2.2.7, the trade data is given in a CashPositionData sub node.
- Derivative: An arbitrary underlying derivative trade (of any type covered by ORE), allowing the set up of a so called Portfolio Swap with multiple underlying derivatives. The Derivative subnode has exactly two subnodes:
  - Id: A string with a unique identifier for the derivative position, typically starting with *DERIV:*.
  - Trade: The root node of a derivative trade.
- PortfolioIndexTradeData: This is a Portfolio Swap that references an underlying Basket via the `BasketName` identifier. The underlying basket can have an arbitrary number underlying derivatives of any supported TradeType. The PortfolioIndexTradeData subnode has one subnode:
  - BasketName: A string with a unique identifier for the portfolioIndex, matching the underlying reference basket.
  - IndexQuantity: Number of shares of the index.

Except for PortfolioIndexTradeData, each trade is specified by a `TradeType` and a trade type dependent data block as listed above. Listing 109 shows an example for a convertible bond underlying. Listing 110 shows an example for an equity basket underlying. Listing 111 shows an example for a bond basket underlying. Listing 112 shows an example for a Derivative underlying (with 3 underlying trades in this case). Listing 113 shows an example for a PortfolioIndexTradeData underlying.

- The `ReturnData` block specifies the details of the return leg.
  - Payer: Indicates whether the return leg is paid.

    Allowable values: *true, false*
  - Currency: The currency in which the return is expressed. This can be different from the underlying currency ("composite" swap) and also from the funding leg currency ("cross currency" swap). The "composite" and "cross currency" features can occur alone or in combination.

    Allowable values: A valid currency code, see `Currency` in Table 13, provided it is the same as on the funding leg.

– ScheduleData: The reference schedule for the return leg, where the valuation dates are derived from this schedule using the ObservationLag, ObservationConvention and ObservationCalendar fields. The payment dates are derived from this schedule using the PaymentLag, PaymentConvention and PaymentCalendar fields. The payment dates can also be given as an explicit list in the PaymentDates node.

  Allowable values: A `ScheduleData` block as defined in section 2.3.4

– ObservationLag [Optional]: The lag between the valuation date and the reference schedule period start date.

  Allowable values: Any valid period, i.e. a non-negative whole number, followed by *D* (days), *W* (weeks), *M* (months), *Y* (years). Defaults to *0D* if left blank or omitted.

– ObservationConvention [Optional]: The roll convention to be used when applying the observation lag.

  Allowable values: A valid roll convention (*F, MF, P, MP, U, NEAREST*), see Table 14 Roll Convention. Defaults to *U* if left blank or omitted.

– ObservationCalendar [Optional]: The calendar to be used when applying the observation lag.

  Allowable values: Any valid calendar, see Table 17 Calendar. Defaults to the *NullCalendar* (no holidays) if left blank or omitted.

– PaymentLag [Optional]: The lag between the reference schedule period end date and the payment date.

  Allowable values: Any valid period, i.e. a non-negative whole number, optionally followed by *D* (days), *W* (weeks), *M* (months), *Y* (years). Defaults to *0D* if left blank or omitted. If a whole number is given and no letter, it is assumed that it is a number of *D* (days).

– PaymentConvention [Optional]: The business day convention to be used when applying the payment lag.

  Allowable values: A valid roll convention (*F, MF, P, MP, U, NEAREST*), see Table 14 Roll Convention. Defaults to *U* if left blank or omitted.

– PaymentCalendar [Optional]: The calendar to be used when applying the payment lag.

  Allowable values: Any valid calendar, see Table 17 Calendar. Defaults to the *NullCalendar* (no holidays) if left blank or omitted.

– PaymentDates [Optional]: This node allows for the specification of a list of explicit payment dates, using `PaymentDate` elements. The list must contain exactly $n - 1$ dates where n is the number of dates in the reference schedule given in the ScheduleData node. See Listing 107 for an example with an assumed ScheduleData with 4 dates.

Listing 107: Payment dates

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

– InitialPrice [Optional]: The equity (or bond) price of the underlying on the valuation date associated with the start date. Commonly contractually given. The price can be given in the underlying currency or the return currency as specified by the InitialPriceCurrency field and is given as

  * a (dirty) price for Bond, ForwardBond and Convertible Bond underlyings, the format is dependent on the price quotation method of the referenced bond:

    · Percentage of Par: the InitialPrice should be given as e.g. 1.02 for 102% relative dirty price

    · Currency per Unit: the InitialPrice should be given as e.g. 0.51 for a dirty amount of 51 USD per unit of the bond worth (say) 50.0 USD.

  * the weighted price of one unit of the bond underlying basket, notice that this is always a "percentage of par" price regardless of the quotation style of the single bonds in the basket

  * the (weighted) price of (one unit of) the equity underlying (basket)

  * the (weighted) price of (one unit of) the equity option underlying (basket)

  * an *absolute amount in the initial price ccy ("dollar amount")* if more than one underlying is specified and if a derivative is specified

  * absolute NPV if underlying is a CBO

  Notice that for an equity basket underlying with several currencies involved, the initial price is assumed to be given in the return currency in case no InitialPriceCurrency is given.

  Allowable values: A real number. If omitted or left blank it defaults to the equity (or bond) price of the valuation date associated with the start date. When this valuation date is in the future there is no fixed price, and in these cases the InitialPrice defaults to the forward price.

– InitialPriceCurrency [Optional]: Only relevant if InitialPrice is given. This specifies whether the initial price is given in the asset currency, the return currency or the funding currency.

  Allowable values: One of the currencies in ReturnData / Currency (return currency), FundingData/ LegData / currency (funding currency) or the currency of the underlying asset. Defaults to the return currency if omitted.

– FXTerms [Mandatory when underlying asset / return / additional cashflow / funding currencies differ]: If the underlying asset currency is different from the return currency, an FXIndex for the conversion underlying / return currency must be given. The same holds for the funding and additional cashflow currencies: Whenever one of these currencies are different from the underlying currency, an FXIndex for the conversion to the underlying currency must be given. If multiple currencies differ, multiple FXIndex elements must be given.

  * FXIndex: The fx index to use for the conversion, this must contain the funding / return / additional cashflow currency and the underlying asset currency (in the order defined in table 21, i.e. it does not matter which one is the funding / return / additional cashflow currency and which is the underlying currency)

    Allowable values: see 21

  Notice that for an underlying of type EquityPosition or EquityOptionPosition additional `FXIndex` entries are required if there is more than one equity position in a different currency: Eventually, for each equity currency there must be a `FXIndex` specifying the conversion from the equity currency to the funding currency (or for the return/cashflow vs funding currency conversion). In this case multiple `FXIndex` entries are used within a single `FXTerms` node, see 108.

Listing 108: FXTerms with multiple FXIndex

```
<FXTerms>
      <FXIndex>FX-TR20H-GBP-SEK</FXIndex>
      <FXIndex>FX-TR20H-GBP-EUR</FXIndex>
      <FXIndex>FX-TR20H-GBP-USD</FXIndex>
</FXTerms>
```

– PayUnderlyingCashFlowsImmediately [Optional]: If true, underlying cashflows like coupon or amortisation payments from bonds or dividend payments from equities, are paid when they occur. If false, these cashflows are paid together with the next return payment. If omitted, the default value is false for trade type TotalReturnSwap and true for trade type ContractForDifference.

  Allowable values: true (immediate payment of underlying cashflwos) or false (underlying cashflows are paid on the next return payment date)

- The `FundingData` block specifies the details of the funding leg(s). The block is optional and can be omitted if no funding legs are present in the swap (e.g. for CFDs). It contains one or more LegData nodes, see 2.3.3. Allowed leg types are
  – Fixed
  – Floating
  – CMS

  - CMB

  The number of coupons defined by the legs often match the number of periods of the return schedule, but this is not a strict requirement. All funding legs must share the same payment currency.

  There are several ways to determine the notional of each funding leg, which is determined by additional, optional NotionalType tags. If given, there must be exactly one NotionalType tag for each LegData nodes. The types have the following meanings:

  - "PeriodReset": the notional of a funding period is determined by the underlying price on the last valuation date before or on the accrual start date of the relevant funding coupon, this price is converted to the funding currency using the FX rate on this same valuation date for compo / cross currency swaps.
  - "DailyReset": the notional of a funding period is determined by the underlying price on each day of the accrual period, again converted to the funding currency using the FX rate of the same date for compo / cross currency swaps. This notional type is only supported for fixed rate funding legs.
  - "Fixed": The notional is explicitly given in the leg data.

  If the NotionalType tags are not given, they default to "PeriodReset" in case no explicit notional is given on the leg and "Fixed" in case an explicit notional is given on the leg. See listing 109 for and example with two funding legs, one with a notional of type DailyReset and one with a notional of type PeriodReset.

  If a FundingResetGracePeriod is given, a lag of the given number of calendar days is applied when determining the relevant return valuation date that determines the funding notional. For example if FundingResetGracePeriod is set to 2, a valuation date that lies at most 2 calendar days after the funding accrual start date will be still considered eligible for this period.

- The `AdditionalCashflowData` block is optional and specifies unpaid amounts to be included in the NPV. The type of this leg must be Cashflow. The currency of the leg must be either the asset currency or the funding currency or the return currency.

Listing 109: Generic Total Return Swap with Convertible Bond underlying

```
<TotalReturnSwapData>
  <UnderlyingData>
    <Trade>
      <TradeType>Bond</TradeType>
      <BondData>
        <SecurityId>ISIN:XY1000000000</SecurityId>
        <BondNotional>1000000.00</BondNotional>
      </BondData>
    </Trade>
  </UnderlyingData>
  <ReturnData>
    <Payer>false</Payer>
    <Currency>EUR</Currency>
    <ScheduleData>...</ScheduleData>
    <ObservationLag>0D</ObservationLag>
    <ObservationConvention>P</ObservationConvention>
    <ObservationCalendar>USD</ObservationCalendar>
    <PaymentLag>2D</PaymentLag>
    <PaymentConvention>F</PaymentConvention>
    <PaymentCalendar>TARGET</PaymentCalendar>
    <!-- <PaymentDates> -->
    <!--   <PaymentDate> ... </PaymentDate> -->
    <!--   <PaymentDate> ... </PaymentDate> -->
    <!-- </PaymentDates> -->
    <InitialPrice>1.05</InitialPrice>
    <InitialPriceCurrency>EUR</InitialPriceCurrency>
    <FXTerms>
      <FXIndex>FX-ECB-EUR-USD</FXIndex>
      <FXIndex>FX-ECB-GBP-USD</FXIndex>
    </FXTerms>
    <PayUnderlyingCashFlowsImmediately>false</PayUnderlyingCashFlowsImmediately>
  </ReturnData>
  <FundingData>
    <FundingResetGracePeriod>2</FundingResetGracePeriod>
    <NotionalType>DailyReset</NotionalType>
    <LegData>
      <Payer>true</Payer>
      <LegType>Fixed</LegType>
      ...
    </LegData>
    <NotionalType>PeriodReset</NotionalType>
    <LegData>
      <Payer>true</Payer>
      <LegType>Floating</LegType>
      ...
    </LegData>
  </FundingData>
  <AdditionalCashflowData>
    <LegData>
      <Payer>false</Payer>
      <LegType>Cashflow</LegType>
      ...
    </LegData>
  </AdditionalCashflowData>
</TotalReturnSwapData>
```

Listing 110: Generic Total Return Swap with equity basket underlying

```
    <TotalReturnSwapData>
      <UnderlyingData>
        <Trade>
          <TradeType>EquityPosition</TradeType>
          <EquityPositionData>
            <!-- basket price = quantity x sum_i ( weight_i x equityPrice_i x fx_i ) -->
            <Quantity>1000</Quantity>
            <Underlying>
              <Type>Equity</Type>
              <Name>BE0003565737</Name>
              <Weight>0.5</Weight>
              <IdentifierType>ISIN</IdentifierType>
              <Currency>EUR</Currency>
              <Exchange>XFRA</Exchange>
            </Underlying>
            <Underlying>
              <Type>Equity</Type>
              <Name>GB00BH4HKS39</Name>
              <Weight>0.5</Weight>
              <IdentifierType>ISIN</IdentifierType>
              <Currency>GBP</Currency>
              <Exchange>XLON</Exchange>
            </Underlying>
          </EquityPositionData>
        </Trade>
      </UnderlyingData>
      <ReturnData>
        ...
        <InitialPrice>112.0</InitialPrice>
        <InitialPriceCurrency>USD</InitialPriceCurrency>
        <FXTerms>
          <FXIndex>FX-ECB-EUR-USD</FXIndex>
          <FXIndex>FX-TR20H-GBP-USD</FXIndex>
        </FXTerms>
      </ReturnData>
      <FundingData>
        <LegData>
          <Payer>true</Payer>
          <LegType>Floating</LegType>
          <Currency>USD</Currency>
          ...
        </LegData>
      </FundingData>
      <AdditionalCashflowData>
        <LegData>
          <Payer>false</Payer>
          <LegType>Cashflow</LegType>
          ...
        </LegData>
      </AdditionalCashflowData>
    </TotalReturnSwapData>
  </Trade>
```

Listing 111: Generic Total Return Swap with bond basket underlying

```
    <TotalReturnSwapData>
      <UnderlyingData>
        <Trade>
          <TradeType>BondPosition</TradeType>
          <BondBasketData>
            <Quantity>100000000</Quantity>
            <Identifier>ISIN:GB00B4KT9Q30</Identifier>
          </BondBasketData>
        </Trade>
      </UnderlyingData>
      <!-- omitting ReturnData, FundingData, AdditionalCashflowData -->
    </TotalReturnSwapData>
  </Trade>
```

Listing 112: Generic Total Return Swap on a Derivative underlying

```
    <TotalReturnSwapData>
      <UnderlyingData>
        <Derivative>
          <Id>DERIV:TEST1</Id>
          <Trade>
            <TradeType>Swaption</TradeType>
              <SwaptionData> ... </SwaptionData>
          </Trade>
        </Derivative>
        <Derivative>
          <Id>DERIV:TEST2</Id>
          <Trade>
            <TradeType>Swap</TradeType>
              <SwapData> ... </SwapData>
          </Trade>
        </Derivative>
         <Derivative>
          <Id>DERIV:TEST3</Id>
          <Trade>
            <TradeType>EquityPosition</TradeType>
              <EquityPositionData> ... </EquityPositionData>
          </Trade>
        </Derivative>
      </UnderlyingData>
      <!-- omitting ReturnData, FundingData, AdditionalCashflowData -->
    </TotalReturnSwapData>
  </Trade>
```

Listing 113: Generic Total Return Swap on a PortfolioIndexTradeData underlying

```
    <TotalReturnSwapData>
      <UnderlyingData>
        <PortfolioIndexTradeData>
          <BasketName>MSFDSJP</BasketName>
        </PortfolioIndexTradeData>
      </UnderlyingData>
      <!-- omitting ReturnData, FundingData, AdditionalCashflowData -->
    </TotalReturnSwapData>
  </Trade>
```

Listing 114: Generic Total Return Swap on a commodity index underlying

```
    <TotalReturnSwapData>
      <UnderlyingData>
        <Trade>
          <TradeType>CommodityPosition</TradeType>
          <CommodityPositionData>
            <!-- basket price = quantity x sum_i ( weight_i x price_i x fx_i ) -->
            <Quantity>1000</Quantity>
            <Underlying>
              <Type>Commodity</Type>
              <Name>RIC:.BCOM</Name>
              <Weight>1.0</Weight>
              <PriceType>Spot</PriceType>
            </Underlying>
          </CommodityPositionData>
        </Trade>
      </UnderlyingData>
      <!-- omitting ReturnData, FundingData, AdditionalCashflowData -->
    </TotalReturnSwapData>
  </Trade>
```

Listing 115: CFD on STOXX50E with initial price 3399.20 EUR

```
<ContractForDifferenceData>
  <UnderlyingData>
    <Trade>
      <TradeType>EquityPosition</TradeType>
      <EquityPositionData>
        <Quantity>1000</Quantity>
        <Underlying>
          <Type>Equity</Type>
          <Name>.STOXX50E</Name>
          <Weight>1.0</Weight>
          <IdentifierType>RIC</IdentifierType>
        </Underlying>
      </EquityPositionData>
    </Trade>
  </UnderlyingData>
  <ReturnData>
    <Payer>false</Payer>
    <Currency>EUR</Currency>
    <ScheduleData>
      <Dates>
        <Dates>
          <!-- the start date of the CFD on which the initial price was set -->
          <Date>2018-09-28</Date>
          <!-- fictitious closing date, e.g. set to "tomorrow" -->
          <Date>2019-01-04</Date>
        </Dates>
      </Dates>
    </ScheduleData>
    <InitialPrice>3399.20</InitialPrice>
    <InitialPriceCurrency>EUR</InitialPriceCurrency>
  </ReturnData>
  </ContractForDifferenceData>
```

2.2.69 Equity Outperformance Option

Payoff

An Equity Outperformance option has a payoff that depends on the 'outperformance' of two equity indices (i.e. the difference between their returns) against a strike return. The buyer has the right but not the obligation to receive the outperformance in exchange for the strike rate at a predetermined time in the future.

Payoff:

$$N \cdot \max(0, R_1 - R_2 - K)$$

where N is the notional amount, R is the return of an index, and K is the strike.

A knockIn and knockOut price may be provided. The payoff is then only paid if on the exercise date the price of Index2 is above the knockIn price and below the knockOut price.

The pricing methodology was generalized from [31], 13.16.2.

Input

The `EquityOutperformanceOptionData` node is the trade data container for the *EquityOutperformanceOption* trade type. The `EquityOutperformanceOptionData` node includes one `OptionData` trade component sub-node plus elements specific to the Equity Outperformance Option.

The structure of an example `EquityOutperformanceOptionData` node for an Equity Outperformace Option is shown in Listing 116.

Listing 116: Equity Outperformance Option Data

```
<EquityOutperformanceOptionData>
    <OptionData>
      <LongShort>Long</LongShort>
      <OptionType>Call</OptionType>
      <Style>European</Style>
      <Settlement>Cash</Settlement>
      <ExerciseDates>
        <ExerciseDate>2022-09-21</ExerciseDate>
      </ExerciseDates>
      ...
    </OptionData>
  <Currency>USD</Currency>
  <Notional>500000</Notional>
  <Underlying1>
    <Type>Equity</Type>
    <Name>RIC:.SPX</Name>
  </Underlying1>
  <Underlying2>
    <Type>Equity</Type>
    <Name>RIC:.NDX</Name>
  </Underlying2>
  <InitialPrice1>2140</InitialPrice1>
  <InitialPrice2>13000</InitialPrice2>
  <StrikeReturn>0.01</StrikeReturn>
  <KnockInPrice>12500</KnockInPrice>
  <KnockOutPrice>14000</KnockOutPrice>
</EquityOutperformanceOptionData>
```

The Payoff is:

$$N \cdot \max(0, R_1 - R_2 - K)$$

where:

- N is the notional amount
- R_1 is the return of `Underlying1`
- R_2 is the return of `Underlying2`
- K is the `StrikeReturn`.

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

- OptionData: This is a trade component sub-node outlined in section 2.3.1. The relevant fields in the `OptionData` node for an EquityOutperformanceOption are:

- `LongShort` The allowable values are *Long* or *Short*.
- `OptionType` The allowable values are *Call* or *Put*. *Call* means that the holder has the right but not obligation to receive the Outperformance and pay the StrikeReturn. *Put* means that the holder has the right but not obligation to pay the Outperformance and receive the StrikeReturn.
- `Style` The allowable value is *European*. Note that the Equity Outperformance Option type allows for *European* option style only.
- `Settlement` The allowable values are *Cash* or *Physical*.
- An `ExerciseDates` node where exactly one ExerciseDate date element must be given.
- Premiums [Optional]: Option premium amounts paid by the option buyer to the option seller.

  Allowable values: See section 2.3.2

- Currency: The currency of the equity outperformance option.

  Allowable values: See Table 15 `Currency`.

- Underlying1: Specifies the first underlying equity. This in turn defines the equity curve used for pricing. The `Underlying` node is described in further detail in Section 2.3.29. Note that the node name is `Underlying1`.
- Underlying2: Specifies the second underlying equity. This in turn defines the equity curve used for pricing. The `Underlying` node is described in further detail in Section 2.3.29. Note that the node name is `Underlying2`.

  Also note that the equities in Underlying1 and Underlying2 must be quoted in the same currency.

- InitialPrice1: Specifies the initial price for first underlying equity.

  Allowable values: Any positive real number.

- InitialPrice2: Specifies the initial price for second underlying equity.

  Allowable values: Any positive real number.

- StrikeReturn: The option strike return.

  Allowable values: Any positive real number.

- Notional: The notional amount for the trade.

  Allowable values: Any positive real number.

- KnockInPrice[Optional]: The payoff is only paid if on the settlement date the price of underlying2 is above this value.

  Allowable values: Any positive real number.

- KnockOutPrice[Optional]: The payoff is only paid if on the settlement date the price of underlying2 is below this value.

  Allowable values: Any positive real number.

- InitialPriceCurrency1 [Optional]: Only relevant if InitialPrice1 is given in a currency other than Underlying1's currency.

  Allowable values: See Table 15 `Currency`.

- InitialPriceCurrency2 [Optional]: Only relevant if InitialPrice1 is given in a currency other than Underlying2's currency.

  Allowable values: See Table 15 `Currency`.

- InitialPriceFXTerms1 [Mandatory when InitialPriceCurrency1 is provided]: The node must be given if and only if the underlying currency is different from the initialPrice currency. The node contains the following sub nodes:

  – FXIndex: The fx index to use for the conversion, this must contain the underlying asset currency and the funding leg currency (in the order defined in table 21, i.e. it does not matter which one is the asset currency and which is the funding currency)

  Allowable values: see 21

  – InitialPriceFXTerms2 [Mandatory when InitialPriceCurrency2 is provided]: The node must be given if and only if the underlying currency is different from the initialPrice currency. Contains the same subnodes as InitialPriceFXTerms1.

  Allowable values: Any valid calendar, see Table 17 Calendar. Defaults to the *NullCalendar* (no holidays) if left blank or omitted.

2.2.70 Double Digital Option

Payoff

A double digital option is a binary option that pays out a fixed amount if the spots of two FX, Equtiy, Commodity underlyings are simultaneously in the money w.r.t. given strikes and option types at the option expiry. As an example consider the following term sheet data:

- Expiry: 2021-09-01
- Settlement: 2021-09-03
- Binary Payout: 12000000
- BinaryLevel1: 1.1
- BinaryLevel2: 0.007
- Type1: Call
- Type2: Put
- Position: Long
- Underlying1: FX-ECB-EUR-USD
- Underlying2: FX-ECB-JPY-USD
- PayCcy: USD

This trade pays 12000000 USD to the option holder on the settlement date 2021-09-03 if the EUR-USD FX spot rate is above 1.1 *and* the JPY-USD FX spot rate is below 0.007 on the expiry date 2021-09-01.

Input

The `DoubleDigitalOptionData` node is the trade data container for the DoubleDigitalOption trade type, listing 117 shows the structure of an example with two underlying FX rates. Equity, Commodity and IR underlyings are also supported in arbitrary combinations. A double digital option is a binary option that pays out a fixed amount if the two underlyings (FX spots, Equity or Commodity prices, interest rates) are simultaneously in the money w.r.t. given strikes and option types at the option expiry.

Listing 117: Double Digital Option data

```
<DoubleDigitalOptionData>
  <Expiry>2021-09-01</Expiry>
  <Settlement>2021-09-03</Settlement>
  <BinaryPayout>12000000</BinaryPayout>
  <BinaryLevel1>1.1</BinaryLevel1>
  <BinaryLevel2>0.006</BinaryLevel2>
  <BinaryLevelUpper2>0.008</BinaryLevelUpper2>
  <Type1>Call</Type1>
  <Type2>Collar</Type2>
  <Position>Long</Position>
  <Underlying1>
    <Type>FX</Type>
    <Name>ECB-EUR-USD</Name>
  </Underlying1>
  <Underlying2>
    <Type>FX</Type>
    <Name>ECB-JPY-USD</Name>
  </Underlying2>
  <PayCcy>USD</PayCcy>
</DoubleDigitalOptionData>
```

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

- Expiry: The expiry date of the option. Allowable values are valid dates.
- Settlement: The payout settlement date. Allowable values are valid dates.
- BinaryPayout: The amount that is paid if the option is in the money. Allowable values are all non-negative numbers.
- BinaryLevel1: The strike for underlying 1 for *Call* or *Put* option and the lower bound for a *Collar* option. For an FX underlying (SOURCE-CCY1-CCY2) this is the number of units of CCY2 per units of CCY1. For an Equity underlying this is the equity price expressed in the equity ccy. For a Commodity underlying this is the commodity price expressed in the commodity ccy. For an IR underlying this is the rate expressed in decimal form. Allowable values are non-negative numbers.

- BinaryLevel2: The strike for underlying 2 for *Call* or *Put* option and the lower bound for a *Collar*. For an FX underlying (SOURCE-CCY1-CCY2) this is the number of units of CCY2 per units of CCY1. For an Equity underlying this is the equity price expressed in the equity ccy. For a Commodity underlying this is the commodity price expressed in the commodity ccy. For an IR underlying this is the rate expressed in decimal form. Allowable values are non-negative numbers.
- Type1: The option type that applies to underlying 1. Allowable values: *Call*, *Put* or *Collar*. Underlying 1 is considered to be in the money if the spot is above (Call) / below (Put) the BinaryLevel1 resp. between (Collar) the BinaryLevel1 and BinaryLevelUpper1 at the expiry.
- Type2: The option type that applies to underlying 2. Allowable values: *Call*, *Put* or *Collar*. Underlying 2 is considered to be in the money if the spot is above (Call) / below (Put) the BinaryLevel1 resp. between (Collar) the BinaryLevel2 andthe BinaryLevelUpper2 at the expiry.
- Position: The option position type. Allowable values: *Long* or *Short*.
- Underlying1: The first underlying, see 2.3.29.
- Underlying2: The second underlying, see 2.3.29. Note that Type for both underlyings has allowable values *Equity*, *Commodity*, *FX*, and *IR*.
- Underlying3 [Optional]: If defined, the first underlying in this transaction is treated as a spread between Underlying1 and Underlying3 (i.e. Underlying1 fixing minus Underlying3 fixing), see 2.3.29. Underlying3 Type must be the same as Underlying1 Type.
- Underlying4 [Optional]: If defined, the second underlying in this transaction is treated as a spread between Underlying2 and Underlying4 (i.e. Underlying2 fixing minus Underlying4 fixing), see 2.3.29. Underlying4 Type must be the same as Underlying2 Type.
- PayCcy: The currency in which the `BinaryPayout` is paid. See Table 15 for allowable currency codes.
- BinaryLevelUpper1 [Optional]: This is field is used only for Collar option. The upper bound for underlying 1. For an FX underlying (SOURCE-CCY1-CCY2) this is the number of units of CCY2 per units of CCY1. For an Equity underlying this is the equity price expressed in the equity ccy. For a Commodity underlying this is the commodity price expressed in the commodity ccy. For an IR underlying this is the rate expressed in decimal form. Allowable values are non-negative numbers.
- BinaryLevelUpper2 [Optional]: This field is used only for Collar option. The upper bound for underlying 2. For an FX underlying (SOURCE-CCY1-CCY2) this is the number of units of CCY2 per units of CCY1. For an Equity underlying this is the equity price expressed in the equity ccy. For a Commodity underlying this is the commodity price expressed in the commodity ccy. For an IR underlying this is the rate expressed in decimal form. Allowable values are non-negative numbers.

2.2.71 European Option Contingent on a Barrier

Payoff

This product is a plain vanilla European option on a single underlying (FX, Equity, Commodity or InterestRate) with a knock-in/knock-out feature on another underlying asset. The barrier can be continuously monitored (American) or only at the option expiry date (European). For a knock-in (knock-out) barrier, the option is valid when the price of the underlying barrier moves inside (outside) the barrier, otherwise the payoff is zero. For a European-style barrier, we support FX, Equity, Commodity and InterestRate underlyings. We do not support InterestRate for American-style barriers.

For example, for a European option with a continuous down-in barrier, the option payoff will only apply if the barrier underlying price was less than or equal to the agreed barrier level at any point over the life of the trade. For a continuous up-out barrier, the option will be 'knocked-out' if the barrier underlying price is greater than or equal to the agreed barrier level at any point over the life of the trade, and the final option payoff will be zero.

Input

The trade container for this product is the `EuropeanOptionBarrierData` node, and the corresponding trade type is EuropeanOptionBarrier. The barrier can be continuously monitored (American) or only be monitored on the option expiry date (European). Currently, we support Equity, FX, Commodity and InterestRate for the option underlying. For the barrier underlying, we support Equity, FX, Commodity and InterestRate for European-style barriers, but not InterestRate for an American-style barrier.

Listing 118 shows the structure of an Equity European option with an American-style FX barrier. For a European-style barrier, the `BarrierType` must be set to *European* and the `BarrierSchedule` can be omitted.

Listing 118: European Option Barrier data (continuous barrier)

```
<Trade id="Equity_EuropeanOptionWithAmericanFxBarrier">
  <TradeType>EuropeanOptionBarrier</TradeType>
  <Envelope>
    .....
  </Envelope>
  <EuropeanOptionBarrierData>
    <Quantity>8523</Quantity>
    <PutCall>Call</PutCall>
    <LongShort>Short</LongShort>
    <Strike>3520</Strike>
    <PremiumAmount>114.40</PremiumAmount>
    <PremiumCurrency>EUR</PremiumCurrency>
    <PremiumDate>2019-12-13</PremiumDate>
    <OptionExpiry>2020-06-19</OptionExpiry>
    <OptionUnderlying>
      <Type>Equity</Type>
      <Name>RIC:.STOXX50E</Name>
    </OptionUnderlying>
    <BarrierUnderlying>
      <Type>FX</Type>
      <Name>ECB-EUR-USD</Name>
    </BarrierUnderlying>
    <BarrierLevel>1.09335</BarrierLevel>
    <BarrierType>DownAndIn</BarrierType>
    <BarrierStyle>American</BarrierStyle>
    <BarrierSchedule>
      <Rules>
        <StartDate>2019-12-11</StartDate>
        <EndDate>2020-06-19</EndDate>
        <Tenor>1D</Tenor>
        <Calendar>USA</Calendar>
        <Convention>Following</Convention>
        <TermConvention>Following</TermConvention>
        <Rule>Forward</Rule>
      </Rules>
    </BarrierSchedule>
    <SettlementDate>2020-06-24</SettlementDate>
    <PayCcy>USD</PayCcy>
  </EuropeanOptionBarrierData>
</Trade>
```

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

- Quantity: Number of option contracts.
  Allowable values: Any non-negative number.
- PutCall: Option type.
  Allowable values: *Call, Put*
- LongShort: Own party position.
  Allowable values: *Long, Short*
- Strike: Option strike price.
  Allowable values: Any positive number.

- PremiumAmount: Premium amount per option.
  Allowable values: Any number.
- PremiumCurrency: Currency of the option premium.
  Allowable values: See `Currency` in Table 13.
- PremiumDate: The option premium payment date.
  Allowable values: See `Date` in Table 13.
- OptionExpiry: Option expiry date.
  Allowable values: See `Date` in Table 13.
- OptionUnderlying: The option underlying.
  Allowable values: A node of the same form as `Underlying`, (see 2.3.29). The supported types are *Equity*, *FX*, *Commodity* and *InterestRate*.
- BarrierUnderlying: The underlying monitored against the barrier level.
  Allowable values: A node of the same form as `Underlying`, (see 2.3.29). The supported types are *Equity*, *FX*, *Commodity* and *InterestRate* if `BarrierStyle` is *European*. For `BarrierStyle` *American*, we only support *Equity*, *FX* and *Commodity*.
- BarrierLevel: Knock-in/knock-out barrier level.
  Allowable values: Any number.
- BarrierType: The type of knock-in or knock-out barrier.
  Allowable values: *DownIn, UpIn, DownOut, UpOut*
- BarrierStyle: Whether the barrier is continuously monitored or only at the option expiry date.
  Allowable values: *American, European*
- BarrierSchedule [Optional]: The schedule specifying the schedule of trading days over which the continuous barrier will be monitored. This is required only when `BarrierStyle` is *American*.
  Allowable values: A node of the same form as `ScheduleData`, (see 2.3.4).
- SettlementDate: Settlement date of the option exercise payoff.
  Allowable values: See `Date` in Table 13. The Settlement date must be on or after the Option expiry date.
- PayCcy: Settlement currency.
  Allowable values: See Table 15 `Currency`.

2.2.72 Autocallable Type 01

Payoff

The autocallable option of type 01 is characterized by the following data

- a notional amount N
- a determination level D
- a trigger level T
- The referecne underlying U

- a number of fixing dates f_i for $i = 1, \ldots, n$
- a number of settlement dates s_i for $i = 1, \ldots, n$
- a list of accumulation factors a_i for $i = 1, \ldots, n$
- a cap C

On each fixing date f_i for $i = 1, \ldots, n$, if the underlying spot is at or below the trigger level, i.e.

$$U(f_i) <= T$$

the option holder receives an amount

$$N \cdot a_i$$

on the settlement date s_i and the structure terminates on this same date.

If no trigger event occurs on any of the fixing dates and if the underlying spot is above the determination level on the *last* fixing date f_n, i.e.

$$U(f_n) > D$$

then the option holder *pays* an amount

$$N \min(C, (U(f_n) - D))$$

to the counterparty of the transaction. The underlying can be an Equity, FX or Commodity underlying.

Input

The `Autocallable_01` node is the trade data container for the Autocallable_01 trade type, listing 119 shows the structure of an example.

Listing 119: Autocallable Type 01 data

```
<Autocallable01Data>
  <NotionalAmount>12000000</NotionalAmount>
  <DeterminationLevel>11.0</DeterminationLevel>
  <TriggerLevel>9.8</TriggerLevel>
  <Underlying>
    <Type>FX</Type>
    <Name>ECB-EUR-NOK</Name>
  </Underlying>
  <Position>Long</Position>
  <PayCcy>EUR</PayCcy>
  <FixingDates>
    <ScheduleData>
      <Dates>
        <Dates>
          <Date>2018-09-27</Date>
          <Date>2019-09-27</Date>
          <Date>2020-09-27</Date>
          <Date>2021-09-29</Date>
          <Date>2022-09-28</Date>
        </Dates>
      </Dates>
    </ScheduleData>
  </FixingDates>
  <SettlementDates>
    <ScheduleData>
      <Dates>
        <Dates>
          <Date>2018-10-07</Date>
          <Date>2019-10-09</Date>
          <Date>2020-10-07</Date>
          <Date>2021-10-07</Date>
          <Date>2022-10-07</Date>
        </Dates>
      </Dates>
    </ScheduleData>
  </SettlementDates>
  <AccumulationFactors>
    <Factor>0.344</Factor>
    <Factor>0.733</Factor>
    <Factor>0.911</Factor>
    <Factor>1.123</Factor>
    <Factor>1.544</Factor>
  </AccumulationFactors>
  <Cap>1.0</Cap>
</Autocallable01Data>
```

If a trigger event occurs on the i-th fixing date, the option holder receives the following:

$$Payout = \texttt{NotionalAmount} * AccumulationFactor_i.$$

If a trigger event never occurs and the underlying spot at the fixing date f_n is above the `DeterminationLevel`, the option holder pays the following:

$$Payout = \min(\texttt{Cap}, \texttt{Underlying}(f_n) - \texttt{DeterminationLevel}).$$

The meanings and allowable values of the elements in the `Autocallable01Data` node follow below.

- NotionalAmount: The notional amount of the option. Allowable values are non-negative numbers.
- DeterminationLevel: The determination level. For an FX underlying FX-SOURCE-CCY1-CCY2 this is the number of units of CCY2 per units of CCY1. For an EQ underlying this is the equity price expressed in the equity ccy. Allowable values are non-negative numbers.
- TriggerLevel: The trigger level. For an FX underlying FX-SOURCE-CCY1-CCY2 this is the number of units of CCY2 per units of CCY1. For an EQ underlying this is the equity price expressed in the equity ccy. Allowable values are non-negative numbers.
- Underlying: The option underlying, see 2.3.29.
- Position: The option position type. Allowable values: *Long* or *Short*.
- PayCcy: The pay currency of the option. See the appendix for allowable currency codes.
- FixingDates: The fixing date schedule given as a `ScheduleData` subnode, see 2.3.4
- SettlementDates: The settlement date schedule given as a `ScheduleData` subnode, see 2.3.4
- AccumulationFactors: The accumulation factors given as a list of values corresponding to the fixing dates. Allowable values are non-negative numbers.
- Cap: The maximum amount, per unit of notional, payable by the option holder if a trigger event never occurred (i.e. underlying value was never below the `TriggerLevel` on any of the fixing dates) and if the underlying value is greater than the `DeterminationLevel` at the last fixing date. Allowable values are non-negative numbers.

2.2.73 Performance Option Type 01

Payoff

The performace option of type "01" is characterized by the following data

- a notional amount N
- a participation rate q
- a valuation date V and a settlement date S
- a number of underlyings U_i for $i = 1, \ldots, n$
- weights for the underlyings w_i for $i = 1, \ldots, n$

- initial strike prices for the underlyings s_i for $i = 1, \ldots, n$
- an option strike K

On the valuation date the average performance of the underlying basket is computed as

$$P = \max\left(\sum_{i=1}^{n} w_i \left(\frac{U_i(V)}{s_i} - K\right), 0\right)$$

The option holder receives an amount $N \cdot q \cdot P$ on the settlement date S. The underlyings can be Equity, FX or Commodity underlyings.

The above payoff includes the strike in the performance calculation. There is another variant with excluded strike and payoff:

$$P = \max\left(\left[\sum_{i=1}^{n} w_i \frac{U_i(V)}{s_i}\right] - K, 0\right)$$

Input

The `PerformanceOption_01` node is the trade data container for the PerformanceOption_01 trade type, listing 120 shows the structure of an example.

Listing 120: Performance Option Type 01 data

```
<PerformanceOption01Data>
  <NotionalAmount>12500000</NotionalAmount>
  <ParticipationRate>0.9</ParticipationRate>
  <ValuationDate>2022-05-03</ValuationDate>
  <SettlementDate>2022-05-05</SettlementDate>
  <Underlyings>
    <Underlying>
      <Type>FX</Type>
      <Name>ECB-CHF-EUR</Name>
      <Weight>0.34</Weight>
    </Underlying>
    <Underlying>
      <Type>FX</Type>
      <Name>ECB-NOK-EUR</Name>
      <Weight>0.32</Weight>
    </Underlying>
    <Underlying>
      <Type>FX</Type>
      <Name>ECB-SEK-EUR</Name>
      <Weight>0.24</Weight>
    </Underlying>
    <Underlying>
      <Type>FX</Type>
      <Name>ECB-SEK-EUR</Name>
      <Weight>0.10</Weight>
    </Underlying>
  </Underlyings>
  <StrikePrices>
    <StrikePrice>0.910002</StrikePrice>
    <StrikePrice>0.097192</StrikePrice>
    <StrikePrice>0.096085</StrikePrice>
    <StrikePrice>0.035032</StrikePrice>
  </StrikePrices>
  <Strike>1.15</Strike>
  <StrikeIncluded>true</StrikeIncluded>
  <Position>Long</Position>
  <PayCcy>EUR</PayCcy>
</PerformanceOption01Data>
```

The meanings and allowable values of the elements in the `PerformanceOption01Data` node follow below.

- NotionalAmount: The notional amount of the option. Allowable valus are non-negative numbers.
- ParticipationRate: The participation rate. Allowable values are non-negative numbers. Usually the value will be between 0 and 1.
- ValuationDate: The valuation date. Allowable values are valid dates.
- SettlementDate: The settlement date. Allowable values are valid dates.
- Underlyings: The underlyings of the option. See 2.3.29 for each underlying.
- StrikePrices: The initial strike prices of the underlyings. For an FX underlying

FX-SOURCE-CCY1-CCY2 this is the number of units of CCY2 per units of CCY1. For an EQ underlying this is the equity price expressed in the equity ccy. For a Commodity underlying this is the commodity price quoted as per the underlying commodity. Allowable values are non-negative numbers.

- Strike: The option strike. This is expressed in terms of the performance of the underlying basket (see the product description for more details). Allowable values are numbers.
- StrikeIncluded [optional]: If true the strike is included in the performance calculation, this is also the default if the flag not given. If false the strike is excluded.
- Position: The option position. Allowable values are *Long* or *Short*.
- PayCcy: The payment currency of the option. See the appendix for allowable currency codes.

2.2.74 Window Barrier Option

Payoff

Window Barrier Options pay a (European) vanilla call / put. An American Knock-In or Knock-Out barrier condition can be defined that are monitored continuously between a start and end date. The different barrier types are "UpAndIn", "UpAndOut", "DownAndIn", "DownAndOut".

Input

The `FxWindowBarrierOptionData`, `EquityWindowBarrierOptionData`, `CommodityWindowBarrierOptionData` is the trade data container for the FxWindowBarrierOption, EquityWindowBarrierOption, CommodityBarrierOption trade type. The following listing shows the structure of an example trade for an FX underlying.

```
<Trade id="FX_Window_BarrierOption">
  <TradeType>FxWindowBarrierOption</TradeType>
  <Envelope>
   ...
  </Envelope>
  <FxWindowBarrierOptionData>
    <Currency>USD</Currency>
    <FixingAmount>1000000</FixingAmount>
    <Underlying>
      <Type>FX</Type>
      <Name>ECB-EUR_USD</Name>
    </Underlying>
    <Strike>1.1</Strike>
    <StartDate>2023-03-01</StartDate>
    <EndDate>024-03-01</EndDate>
    <OptionData>
      <LongShort>Long</LongShort>
      <OptionType>Call</OptionType>
      <ExerciseDates>
        <ExerciseDate>2026-03-01</ExerciseDate>
      </ExerciseDates>
```

```
      <Premiums>
        <Premium>
          <Amount>10900</Amount>
          <Currency>EUR</Currency>
          <PayDate>2018-03-01</PayDate>
        </Premium>
      </Premiums>
      <PaymentData>
        <Dates>
          <Date>2026-03-01</Date>
        </Dates>
      </PaymentData>
    </OptionData>
    <BarrierData>
      <Type>UpAndOut</Type>
      <Levels>
        <Level>1.3</Level>
      </Levels>
    </BarrierData>
  </FxWindowBarrierOptionData>
</Trade>
```

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

- Currency: The payout currency. The result of the payout formula above is treated to be in this currency. Note that for (non-quanto) FxWindowBarrierOption this should be the domestic (`CCY2`) currency. For non-quanto Equity- and CommodityWindowBarrierOptions this should be the currency the equity or commodity is quoted in. Notice section "Payment Currency" in ore/Docs/ScriptedTrade.
  Allowable values: See Table 15 `Currency`.

- FixingAmount: The notional amount. For FxWindowBarrierOptions: The FixingAmount is expressed in the foreign currency (`CCY1`). For EquityWindowBarrierOptions: The FixingAmount is expressed as number of shares/units of the underlying equity or equity index. For CommodityWindowBarrierOptions: The FixingAmount is expressed as number of units of the underlying commodity.
  Allowable values: Any positive real number.

- Underlying: A node with the underlying of the Window Barrier instrument. For FxWindowBarrierOption: `Type` is set to *FX* and `Name` is a string of the form *SOURCE-CCY1-CCY2* where `CCY1` is the foreign currency, `CCY2` is the domestic currency, and *SOURCE* is the fixing source, see Table 21 and 2.3.29. For EquityWindowBarrierOption: `Type` is set to *Equity* and `Name` and other fields are as outlined in 2.3.29. For CommodityWindowBarrierOption: `Type` is set to *Commodity* and `Name` is an identifier of the commodity as outlined in 2.3.29 and in Table 25.
  Allowable values: Any FX, Equity or Commodity underlying as specified in 2.3.29

- StrikeData [Optional]: A node containing the strike in `Value` and the currency in which both the underlying and the strike are quoted in `Currency`. Only supported for EquityWindowBarrierOption.
  Allowable values: See `Currency` in Table 13. The strike may be any positive real

number. The currency provided in this node may be quoted as corresponding minor currency to the underlying major currency.

- Strike [Optional]: For FxWindowBarrierOptions: The the fx strike rate is defined as amount in domestic currency (`CCY2`) for one unit of foreign currency (`CCY1`). For Equity- and CommodityWindowBarrierOptions: The strike value for one unit/share/contract of the underlying equity or commodity, expressed in the currency the equity or commodity is quoted in. For EquityWindowBarrierOptions, the `StrikeData` node (see above) should be used. Allowable values: Any positive real number.
- StartDate: The start date of the continuous observation window. Allowable values: Any valid date.
- EndDate: The end date of the continuous observation window. Allowable values: Any valid date.
- OptionData: See 2.3.1. The relevant fields in the `OptionData` node for a Window Barrier Option are:
  - `LongShort` The allowable values are *Long* or *Short*.
  - `OptionType` The allowable values are *Call* or *Put*.
  - `ExerciseDates` Must contain the single exercise date of the option.
  - `Premiums` [Optional] Option premium amounts paid by the option buyer to the option seller. See section 2.3.2
  - `PaymentData` [Optional] Optional pay date associated to the exercise date.
- BarrierData: Specification of the barrier type and level. The barrier is continuosly monitored between the StartDate and EndDate. Allowable values: See 2.3.31, allowable types are *UpAndOut*, *DownAndOut*, *UpAndIn*, *DownAndIn*. Exactly one level must be given as a positive real number.

Window Barrier Options can be alternatively represented as *scripted trades*, refer to ore/Docs/ScriptedTrade for an introduction.

Trade input and the payoff script are described below.

```
<Trade id="WindowBarrierOption#1">
  <TradeType>ScriptedTrade</TradeType>
  <Envelope/>
  <WindowBarrierOptionData>
    <PremiumPayDate type="event">2018-03-01</PremiumPayDate>
    <Settlement type="event">2026-03-01</Settlement>
    <Expiry type="event">2026-03-01</Expiry>
    <StartDate type="event">2023-03-01</StartDate>
    <EndDate type="event">2024-03-01</EndDate>
    <PremiumAmount type="number">10900</PremiumAmount>
    <BarrierLevel type="number">1.3</BarrierLevel>
    <PutCall type="optionType">Call</PutCall>
    <BarrierType type="barrierType">UpOut</BarrierType>
    <LongShort type="longShort">Long</LongShort>
    <Quantity type="number">1000000</Quantity>
```

```
    <Strike type="number">1.1</Strike>
    <PayCcy type="currency">USD</PayCcy>
    <PremiumCcy type="currency">EUR</PremiumCcy>
    <Underlying type="index">FX-ECB-EUR-USD</Underlying>
  </WindowBarrierOptionData>
</Trade>
```

The WindowBarrierOption script referenced in the trade above is shown in Listing 121.

Listing 121: Payoff script for a WindowBarrierOption.

```
REQUIRE BarrierType == 1 OR BarrierType == 2 OR BarrierType == 3 OR BarrierType == 4;

NUMBER i, Payoff, TriggerProbability, ExerciseProbability, isUp, currentNotional;

IF BarrierType == 1 OR BarrierType == 3 THEN
  TriggerProbability = BELOWPROB(Underlying, StartDate, EndDate, BarrierLevel);
ELSE
  TriggerProbability = ABOVEPROB(Underlying, StartDate, EndDate, BarrierLevel);
END;

Payoff = Quantity * PutCall * (Underlying(Expiry) - Strike);
IF Payoff > 0.0 THEN
  IF BarrierType == 1 OR BarrierType == 2 THEN
    Option = PAY(Payoff * TriggerProbability, Expiry, Settlement, PayCcy);
    ExerciseProbability = TriggerProbability;
  ELSE
    Option = PAY(Payoff * (1 - TriggerProbability), Expiry, Settlement, PayCcy);
    ExerciseProbability = (1 - TriggerProbability);
  END;
END;

Option = LongShort * (Option - PAY(PremiumAmount, PremiumPayDate, PremiumPayDate, PremiumCcy));
currentNotional = Quantity * Strike;
```

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

- [event] `Expiry`: Option expiry date.
  Allowable values: See `Date` in Table 13.
- [event] `Settlement`: Option settlement date.
  Allowable values: See `Date` in Table 13.
- [event] `StartDate`: Barrier monitoring start date.
  Allowable values: See `Date` in Table 13.
- [event] `EndDate`: Barrier monitoring start date.
  Allowable values: See `Date` in Table 13.
- [number] `Strike`: The option strike price. For FX, this should be the amount of `CCY1` divided by amount of `CCY2` as defined in *Index*.
  Allowable values: Any positive real number.
- [number] `BarrierLevel`: Knock-in or knock-out barrier level. For FX, this should be quoted as amount of `CCY2` per unit `CCY1` as defined in *Index*.
  Allowable values: Any positive real number.

- [barrierType] `BarrierType`: Barrier type.
  Allowable values: *DownIn, UpIn, DownOut, UpOut*
- [optionType] `PutCall`: Option type. For FX, this should be Call if we buy `CCY1` and sell `CCY2`, Put if we buy `CCY2` and sell `CCY1` (with `CCY1`, `CCY2` defined in *Index*).
  Allowable values: *Call, Put*
- [longShort] `LongShort`: *Long* if we buy the option. *Short* if we sell the option.
  Allowable values: *Long, Short*
- [number] `Quantity`: Number of option contracts. For FX, this should be the bought amount.
  Allowable values: Any positive real number.
- [index] `Underlying`: Underlying index.
  Allowable values: See ore/Docs/ScriptedTrade's Index Section for allowable values.
- [currency] `PayCcy`: The payment currency. For FX, where the underlying is provided in the form `FX-SOURCE-CCY1-CCY2` (see Table 21) this should be `CCY2`. If `CCY1` or the currency of the underlying (for EQ and COMM underlyings), this will result in a quanto payoff. Notice section "Payment Currency" in ore/Docs/ScriptedTrade.
  Allowable values: See Table 15 for allowable currency codes. Allowable values: See `Currency` in Table 13.
- [event] `PremiumPayDate`: Option premium payment date.
  Allowable values: See `Date` in Table 13.
- [number] `PremiumAmount`: Option premium amount in *PremiumPayCcy*.
  Allowable values: Any positive real number.
- [currency] `PremiumCcy`: Option premium currency.
  Allowable values: See Table 15 `Currency`.

2.2.75 Generic Barrier Option

Payoff

Generic Barrier Options pay a (European) vanilla call / put, an asset or nothing or a cash or nothing payoff at their maturity. The payoff at maturity can be made conditional on (single) European Knock-In or Knock-Out barrier condition ("Transatlantic Barrier").

In addition, a number of American Knock-In and Knock-Out barrier conditions can be defined that are monitored on a defined schedule, e.g. a daily schedule between a monitoring start and end date. The different barrier types are "UpAndIn", "UpAndOut", "DownAndIn", "DownAndOut". If at least one Knock-In barrier is defined, the final payoff is only activated if at least one Knock-In barrier is touched. If at least one Konck-Out barrier is defined, the final payoff is zero'ed out if at least one Knock-Out barrier is touched. An option that is knocked out, can not knock in again. However, there are different sub-types of Knock-Out barriers

- KoAlways: The option can knock out any time in the monitoring period
- KoAfterKi: The option can only knock out after a Knock-In
- KoBeforeKi: The option can only knock out before a Knock-In

In case the American or the Transatlantic barrier deactivate the final option (i.e. if the option knocks out if a Knock-Out barrier is defined or does not knock in if a Konck-In barrier is defined), a rebate amount might be specified that is received by the option holder to compensate for the unfortunate deactivation of their option:

If only Knock-Out barriers are defined, one can associate different rebate amounts to each of the barriers. Once the option knocks out at a barrier, the associated amount is paid. Furthermore, the timing of the payment can be "atHit" (i.e. immediately when the barrier is hit) or "atExpiry" (i.e. on the settlement date associated to the expiry date of the underlying option).

If only Knock-In barriers or a mixed set of Knock-In and Konck-Out barriers are defined, only a unique reabte amount can be defined that is paid "atExpiry".

In both cases a separate Transatlantic Barrier Rebate amount can be specified that is paid when the American barriers do not deactivate the underlying option, but the additional Transatlantic Barrier does.

Input

Generic Barrier Options are defined using one of the trade types *FxGenericBarrierOption*, *EquityGenericBarrierOption*, *CommodityGenericBarrierOption* depending on the underlying asset class and an associated node FxGenericBarrierOptionData, EquityGenericBarrierOptionData, CommodityGenericBarrierOptionData. Listing 122 shows an example for an FX Underlying. Generic Barrier Option can have one or multiple underlyings. In the case of multiple underlyings, there must be one level per underlying provided in each barrier, see 123. The nodes have the following meaning:

- Underlying: The underlying definition. Note that for FX underlyings the order of the currencies defines the observed underlying value, i.e. for EUR-USD the domestic currency is USD (the observed value is e.g. 1.2 USD per EUR) while for USD-EUR the domestic currency is EUR (the observed value is e.g. 0.8 EUR per USD).

  Allowable Values: See 2.3.29

- Underlyings: An alternative to *Underlying* - only one can be present. Can contain multiple *Underlying* nodes. Allowable Values: A list of *Underlying* nodes, with each node given by 2.3.29

- PayCurrency: The payment currency. This is required for all Payoff Types and is usually

  – the domestic currency if underlying = FX

  – the eq / comm currency if underlying = Equity, Commodity

  But we allow for quanto payoffs as well, i.e.

  - the foreign currency if underlying = FX or also
  - a third currency if underlying = FX and
  - a currency not equal to the equity, commodity currency for these underlying types.

  See payoff description which amount is paid in which currency dependent on the type.

  Allowable Values: See `Currency` in Table 13.

- OptionData: The option descripting, see 2.3.1 Relevant sub nodes are:
  - LongShort (allowable values: *Long* or *Short*)
  - PayoffType, with S = Underlying Value and K = Strike this is:
    - *Vanilla*: $\max(0, S - K)$ for a call or $\max(0, K - S)$ for a put, this is paid in PayCurrency
    - *AssetOrNothing*: S paid in PayCurrency
    - *CashOrNothing*: Amount paid in PayCurrency
  - OptionType: Required for PayoffType = *Vanilla*, *Call* or *Put*.
  - ExerciseDate: The exercise date
  - Premiums [Optional]: Option premiums to be paid unconditionally. See section 2.3.2

- SettlementDate [Optional]: The date on which the option payoff is settled. The SettlementDate is used unadjusted as given. Instead of the SettlementDate, a settlement lag, convention and calendar relative to the ExerciseDate can be specified, see below. If the SettlementDate is given on the other hand, SettlementLag, SettlementCalendar and SettlementConvention must *not* be given.

  Allowable Values: any valid date greater or equal to the exercise date

- SettlementLag [Optional]: Alternative specification of the option settlement date via a lag, see the explanation under SettlementDate. Defaults to 0D if not given.

  Allowable Values: any valid period (1D, 2W, 3M, ...)

- SettlementCalendar [Optional]: Alternative specification of the option settlement date via a lag, see the explanation under SettlementDate. Defaults to the underlying calendar, if not given.

  Allowable values: See Table 17 Calendar.

- SettlementConvention [Optional]: Alternative specification of the option settlement date via a lag, see the explanation under SettlementDate. Defaults to Following if not given.

  Allowable values: See Table 14 Roll Convention.

- Quantity: The option quantity. Required for PayoffType = *AssetOrNothing*, *Vanilla*. For FX this is the amount in foreign currency. For Equity, Commodity this is the number of equities, commodities.

  Allowable Values: any real number

- Strike: Required for PayoffType = *Vanilla*.

  Allowable Values: any real number

- Amount: Required for PayoffType = *CashOrNothing*.

  Allowable Values: any real number

- Barriers. The barrier definition. Subnodes are:
  - ScheduleData [Optional]: the observation schedule for the barrier (see 2.3.4. Instead of the ScheduleData, a daily schedule w.r.t. the underlying calendar can be specified by populating the StartDate and EndDate nodes.
  - StartDate [Optional]: Start date of the observation schedule, see the explanation under ScheduleData.
  - EndDate [Optional]: End date of the observation schedule, see the explanation under ScheduleData.
  - BarrierData [Optional]: a sequence of barrier definitions. See 2.3.31. Relevant fields are:
    * Type: The barrier type (allowed values are *UpAndIn*, *UpAndOut*, *DownAndIn*, *DownAndOut*)
    * Levels: Exactly one barrier level per BarrierData block must be given.
    * Rebate [Optional]: The rebate amount. Defaults to zero. Rebate amounts and currencies can be different across barriers if only "out" barriers are defined, but must be identical as soon as at least one "in" barrier is defined.
    * RebateCurrency [Optional]: The currency in which the rebate is paid. Defaults to PayCurrency.
    * RebatePayTime [Optional]: *atExpiry* or *atHit*. For "in" barriers only *atExpiry* is allowed.
    * StrictComparison [Optional]: *0*, *1*, or *2*. Defaults to *0*. Determines how the barrier is checked as per:

      0: the barrier checks use <=, >= to check In-barriers and <, > to check Out-barriers.

      1: the barrier checks use strict comparison < and > for both In- and Out-barriers.

      2: the barrier checks use strict or equal comparison <= and >= for both In- and Out-barriers.

  - KikoType: Required if both a KI and KO barriers are defined. Allowable values are *KoAlways*, *KoBeforeKi*, *KoAfterKi*.

- TransatlanticBarrier [Optional]: An additional barrier to be checked on option expiry. May contain exactly one BarrierData block with number of levels equal to the number of underlyings or contain same amount of BarrierData blocks with the number of underlyings, exactly one level is allowed in each block for each underlying. The Type (*UpAndIn, UpAndOut, DownAndIn, DownAndOut*), the (unique) Level, StrictComparison and the Rebate are relevant fields.
  The rebate is paid if there is no knock-out from the American barriers and no payoff from the Transatlantic barrier.

Listing 122: Generic Barrier Option data (FX Underlying)

```
<FxGenericBarrierOptionData>
  <Underlying>
    <Type>FX</Type>
    <Name>ECB-EUR-USD</Name>
  </Underlying>
  <PayCurrency>USD</PayCurrency>
  <OptionData>
    <LongShort>Long</LongShort>
    <PayoffType>Vanilla</PayoffType>
    <OptionType>Call</OptionType>
    <ExerciseDates>
      <ExerciseDate>2023-06-06</ExerciseDate>
    </ExerciseDates>
  </OptionData>
  <SettlementDate>2023-06-08</SettlementDate>
  <Quantity>100000000</Quantity>
  <Strike>1.2</Strike>
  <Barriers>
    <ScheduleData>
      <Rules>
        <StartDate>2021-07-10</StartDate>
        <EndDate>2023-06-06</EndDate>
        <Tenor>1D</Tenor>
        <Calendar>TGT,US</Calendar>
        <Convention>F</Convention>
        <TermConvention>F</TermConvention>
        <Rule>Forward</Rule>
      </Rules>
    </ScheduleData>
    <BarrierData>
      <Type>DownAndOut</Type>
      <Levels>
        <Level>1.1</Level>
      </Levels>
      <Rebate>1000000</Rebate>
      <RebateCurrency>USD</RebateCurrency>
      <RebatePayTime>atExpiry</RebatePayTime>
      <StrictComparison>1</StrictComparison>
    </BarrierData>
    <BarrierData>
      <Type>UpAndIn</Type>
      <Levels>
        <Level>1.3</Level>
      </Levels>
      <Rebate>1000000</Rebate>
      <RebateCurrency>USD</RebateCurrency>
      <RebatePayTime>atExpiry</RebatePayTime>
    </BarrierData>
    <KikoType>KoAfterKi</KikoType>
  </Barriers>
  <TransatlanticBarrier>
    <BarrierData>
      <Type>UpAndOut</Type>
      <Levels>
        <Level>1.3</Level>
      </Levels>
      <Rebate>2000000</Rebate>
      <RebateCurrency>USD</RebateCurrency>
      <StrictComparison>1</StrictComparison>
    </BarrierData>
  </TransatlanticBarrier>
</FxGenericBarrierOptionData>
```

Listing 123: Multi Asset Generic Barrier Option data (FX Underlyings)

```
<FxGenericBarrierOptionData>
  <Underlyings>
    <Underlying>
      <Type>FX</Type>
      <Name>ECB-EUR-USD</Name>
    </Underlying>
    <Underlying>
      <Type>FX</Type>
      <Name>ECB-USD-JPY</Name>
    </Underlying>
  </Underlyings>
  <PayCurrency>USD</PayCurrency>
  <OptionData>
    <LongShort>Long</LongShort>
    <PayoffType>Vanilla</PayoffType>
    <OptionType>Call</OptionType>
    <ExerciseDates>
      <ExerciseDate>2023-06-06</ExerciseDate>
    </ExerciseDates>
  </OptionData>
  <Amount>1500000</Amount>
  <SettlementDate>2023-06-08</SettlementDate>
  <Barriers>
    <ScheduleData>
      <Rules>
        <StartDate>2021-07-10</StartDate>
        <EndDate>2023-06-06</EndDate>
        <Tenor>1D</Tenor>
        <Calendar>TGT,US</Calendar>
        <Convention>F</Convention>
        <TermConvention>F</TermConvention>
        <Rule>Forward</Rule>
      </Rules>
    </ScheduleData>
    <BarrierData>
      <Type>DownAndOut</Type>
      <Levels>
        <Level>1.1</Level>
        <Level>125</Level>
      </Levels>
      <Rebate>1000000</Rebate>
      <RebateCurrency>USD</RebateCurrency>
      <RebatePayTime>atExpiry</RebatePayTime>
    </BarrierData>
    <BarrierData>
      <Type>UpAndIn</Type>
      <Levels>
        <Level>1.3</Level>
        <Level>135</Level>
      </Levels>
      <Rebate>1000000</Rebate>
      <RebateCurrency>USD</RebateCurrency>
      <RebatePayTime>atExpiry</RebatePayTime>
    </BarrierData>
    <KikoType>KoAfterKi</KikoType>
  </Barriers>
  <TransatlanticBarrier>
    <BarrierData>
      <Type>UpAndOut</Type>
      <Levels>
        <Level>1.3</Level>
        <Level>135</Level>
```

2.2.76 Best Entry Option

Payoff

The Best Entry Option is an option on a single underlying that has a payoff contingent on the 'best entry' of the underlying index into a specified region on a set of specified 'strike observation dates', which is determined by a single barrier.

On the option expiry date, if the value of the underlying index (denoted '$Index_{Final}$') is greater than or equal to the strike price K, the payoff of the option is given by:

$$\text{Notional} \times \text{Multiplier} \times \min\left(\text{Cap}, \max\left(0, \frac{Index_{Final} - Index_{Initial}}{Index_{Inital}}\right)\right).$$

If $Index_{Final}$ is less than K, the payoff is given by

$$-\text{Notional} \times \frac{K - Index_{Final}}{Index_{Initial}}.$$

Here, $Index_{Initial}$ is defined as

- The level of the underlying index on a pre-determined 'strike date', if the level of the index on any of the strike observation dates is *not* lower than the barrier level,
- The maximum between the 'reset minimum value' and the lowest index level on a strike observation date, if the level of the index does indeed hit the barrier level at least once.

Input

Best Entry Options are defined using one of the trade types *FxBestEntryOption*, *EquityBestEntryOption*, *CommodityBestEntryOption* depending on the underlying asset class and an associated node FxBestEntryOptionData, EquityBestEntryOptionData, CommodityBestEntryOptionData. Listing 124 shows an example for an Equity Underlying. For a more detailed description of the computation of the payoff of this option, please see the product description. The nodes have the following meaning:

- Underlying: The underlying definition. Note that for FX underlyings the order of the currencies defines the observed underlying value, i.e. for EUR-USD the domestic currency is USD (the observed value is e.g. 1.2 USD per EUR) while for USD-EUR the domestic currency is EUR (the observed value is e.g. 0.8 EUR per USD).

  Allowable Values: See 2.3.29

- Currency: The payment currency.

  Allowable Values: See `Currency` in Table 13.

- SettlementDate: The date on which the option payoff is settled. The SettlementDate is used unadjusted as given.

  Allowable Values: any valid date greater or equal to the exercise date.

- Notional: The notional amount.

  Allowable Values: any real number

- Strike: The strike value used to compute the payoff of the option. This value should be provided as a decimal, representing a percentage of the value of $Index_{Initial}$, e.g. a value of $K = 0.6 \implies 0.6 \times Index_{Initial}$ in the computation of the payoff. Allowable Values: any real number

- Multiplier: The payoff multiplier used in the case that the underlying index is greater than the strike on the settlement date. If omitted defaults to 1.

  Allowable Values: any real number

- TriggerLevel: The value that is compared to the underlying index on each strike observation date to determine if a Trigger Event has occurred. This should be provided as a decimal, representing a percentage of the value of Strike Index Level, ie the value of the underlying on the `StrikeDate`.

  Allowable Values: any real number

- LongShort: Denotes whether the payoff is computed relative to the holder or seller of the option.

  Allowable Values: *Long*, *Short*.

- Cap: The maximum value of the payoff (before the notional and multiplier are applied). This value should be interpreted as a percentage and should be in decimal format in the trade XML, e.g. $0.06 = 6\%$.

  Allowable Values: any real number

- ResetMinimum: The minimum value of $Index_{Initial}$ in the case that a Trigger Event has occurred at least once during the option's lifetime. This should be provided as a decimal, representing a percentage of the value of Strike Index Level used in the computation of $Index_{Initial}$.

  Allowable Value: any real number

- StrikeDate: The date on which the level on the underlying index is used to compute the payoff, in the case that there has not been a Trigger Event during the option's lifetime.

  Allowable Value: any valid date before the option expiry date.

- ExpiryDate: The date on which the option expires and the payoff is computed.

  Allowable Values: any valid date before the SettlementDate

- Premium: The option premium. Defaults to 0.

  Allowable Values: any real number.

- PremiumDate: The date on which the option premium is paid.

  Allowable Values: any valid date.

- StrikeObservationDates: The set of dates on which the underlying index level is observed - the lowest of which is used to compute the option payoff if the underlying index is greater than the strike on the expiry date.

  Allowable Values: any valid dates schedule (see 2.3.4).

Listing 124: Best Entry Option data (Equity Underlying)

```
<EquityBestEntryOptionData>
  <LongShort>Long</LongShort>
  <Strike>0.85</Strike>
  <Cap>0.06</Cap>
  <ResetMinimum>0.85</ResetMinimum>
  <Notional>1000000</Notional>
  <Multiplier>1</Multiplier>
  <TriggerLevel>0.95</TriggerLevel>
  <SettlementDate>2021-11-20</SettlementDate>
  <PremiumDate>2021-11-22</PremiumDate>
  <StrikeDate>2020-12-15</StrikeDate>
  <Underlying>
    <Type>Equity</Type>
    <Name>RIC:.SPX</Name>
  </Underlying>
  <StrikeObservationDates>
    <Dates>
      <Dates>
        <Date>2021-03-01</Date>
        <Date>2021-06-01</Date>
        <Date>2021-09-01</Date>
      </Dates>
    </Dates>
  </StrikeObservationDates>
  <Currency>USD</Currency>
  <Premium>100</Premium>
  <ExpiryDate>2021-11-20</ExpiryDate>
</EquityBestEntryOptionData>
```

2.2.77 Basket Options

Basket Options are represented as traditional trades or *scripted trades*, refer to ore/Docs/ScriptedTrade for an introduction of the latter. Each of the supported variations is represented by a separate payoff script as shown in Table 4.

Basket Option Type	Payoff Script Name
Vanilla	VanillaBasketOption
Asian	AsianBasketOption
Average strike	AverageStrikeBasketOption
Lookback call	LookbackCallBasketOption
Lookback put	LookbackPutBasketOption

Table 4: Basket option types and associated script names.

The supported underlying types are Equity, Fx or Commodity resulting in corresponding trade types and trade data container names

- EquityBasketOption / EquityBasketOptionData
- FxBasketOption / FxBasketOptionData
- CommodityBasketOption / CommodityBasketOptionData

Trade input and the associated payoff script are described in the following for all supported Basket Option variations.

Vanilla Basket Option

The traditional trade representation is as follows, using an equity underlying in this example:

```
<Trade id="VanillaBasketOption#1">
  <TradeType>EquityBasketOption</TradeType>
  <Envelope>
    <CounterParty>CPTY_A</CounterParty>
    <NettingSetId>CPTY_A</NettingSetId>
    <AdditionalFields/>
  </Envelope>
  <EquityBasketOptionData>
    <Currency>USD</Currency>
    <Notional>1</Notional>
    <Strike>5000</Strike>
    <Underlyings>
      <Underlying>
        <Type>Equity</Type>
        <Name>RIC:.SPX</Name>
        <Weight>1.0</Weight>
      </Underlying>
      <Underlying>
        <Type>Equity</Type>
        <Name>RIC:.STOXX50E</Name>
        <Currency>EUR</Currency>
        <Weight>1.0</Weight>
      </Underlying>
    </Underlyings>
    <OptionData>
      <LongShort>Long</LongShort>
      <OptionType>Call</OptionType>
      <PayoffType>Vanilla</PayoffType>
      <ExerciseDates>
        <ExerciseDate>2020-02-15</ExerciseDate>
      </ExerciseDates>
      <Premiums> ... </Premiums>
    </OptionData>
    <Settlement>2020-02-20</Settlement>
  </EquityBasketOptionData>
</Trade>
```

with the following elements:

- Currency: The pay currency. Notice section "Payment Currency" in ore/Docs/ScriptedTrade.
  Allowable values: all supported currency codes, see Table 15 `Currency`.
- Notional: The quantity (for equity, commodity underlyings) / foreign amount (fx underlying)
  Allowable values: all positive real numbers
- Strike: The strike of the option
  Allowable values: all positive real numbers

- Underlyings: The basket of underlyings.
  Allowable values: for each underlying see 2.3.29
- OptionData: relevant are the long/short flag, the call/put flag, the payoff type (must be set to Vanilla to identify the payoff), and the exercise date (exactly one date must be given). A *Premiums* node can be added to represent deterministic option premia to be paid by the option holder.
  Allowable values: see 2.3.1 for the general structure of the option data node
- Settlement: the settlement date (optional, if not given defaults to the exercise date)
  Allowable values: each valid date.

The representation as a scripted trade is as follows:

```
<Trade id="VanillaBasketOption#1">
  <TradeType>ScriptedTrade</TradeType>
  <Envelope/>
  <VanillaBasketOptionData>
    <Expiry type="event">2020-02-15</Expiry>
    <Settlement type="event">2020-02-20</Settlement>
    <PutCall type="optionType">Call</PutCall>
    <LongShort type="longShort">Long</LongShort>
    <Notional type="number">1</Notional>
    <Strike type="number">5000</Strike>
    <Underlyings type="index">
      <Value>EQ-RIC:.SPX</Value>
      <Value>EQ-RIC:.STOXX50E</Value>
    </Underlyings>
    <Weights type="number">
      <Value>1.0</Value>
      <Value>1.0</Value>
    </Weights>
    <PayCcy type = "currency">USD</PayCcy>
  </VanillaBasketOptionData>
</Trade>
```

The VanillaBasketOption script referenced in the trade above is shown in Listing 125.

Listing 125: Payoff script for a VanillaBasketOption.

```
REQUIRE SIZE(Underlyings) == SIZE(Weights);

NUMBER u, basketPrice, ExerciseProbability, Payoff, currentNotional;

FOR u IN (1, SIZE(Underlyings)) DO
    basketPrice = basketPrice + Underlyings[u](Expiry) * Weights[u];
END;

Payoff = max(PutCall * (basketPrice - Strike), 0);

Option = LongShort * Notional * PAY(Payoff, Expiry, Settlement, PayCcy);

IF Payoff > 0 THEN
    ExerciseProbability = 1;
END;
currentNotional = Notional * Strike;
```

The meanings and allowable values of the elements in the `VanillaBasketOptionData` node are given below, with data type indicated in square brackets.

- [event] `Expiry`: Option expiry date.
  Allowable values: See `Date` in Table 13.

- [event] `Settlement`: Option settlement date.
  Allowable values: See `Date` in Table 13.
- [optionType] `PutCall`: Option type with
  Allowable values *Call, Put*.
- [longShort] `LongShort`: Position type, *Long* if we buy, *Short* if we sell.
  Allowable values: *Long, Short*.
- [number] `Notional`: Quantity multiplier applied to the basket price
  Allowable values: Any positive real number.
- [number] `Strike`: Strike basket price in PayCcy (see below)
  Allowable values: Any positive real number.
- [index] `Underlyings`: List of underlying indices enclosed by `<Value>` and `</Value>` tags.
  Allowable values: See ore/Docs/ScriptedTrade's Index Section for allowable values.
- [number] `Weights`: List of weights applied to the underlying prices in the basket, given in the same order as the Underlyings above, each weight enclosed by `<Value>` and `</Value>` tags.
  Allowable values: Any positive real number.
- [currency] `PayCcy`: The payment currency. For FX, where the underlying is provided in the form `FX-SOURCE-CCY1-CCY2` (see Table 21) this should be `CCY2`. If `CCY1` or the currency of the underlying (for EQ and COMM underlyings), this will result in a quanto payoff. Notice section "Payment Currency" in ore/Docs/ScriptedTrade. Allowable values: See Table 15 `Currency` for allowable currency codes.

Asian Basket Option

The traditional trade representation is as follows, using an equity underlying in this example:

```
<Trade id="AsianBasketOption#1">
  <TradeType>EquityBasketOption</TradeType>
  <Envelope>
    <CounterParty>CPTY_A</CounterParty>
    <NettingSetId>CPTY_A</NettingSetId>
    <AdditionalFields/>
  </Envelope>
  <EquityBasketOptionData>
    <Currency>USD</Currency>
    <Notional>1</Notional>
    <Strike>5000</Strike>
    <Underlyings>
      <Underlying>
        <Type>Equity</Type>
        <Name>RIC:.SPX</Name>
        <Currency>USD</Currency>
        <Weight>1.0</Weight>
      </Underlying>
      <Underlying>
        <Type>Equity</Type>
        <Name>RIC:.STOXX50E</Name>
        <Weight>1.0</Weight>
      </Underlying>
```

```
    </Underlyings>
    <OptionData>
      <LongShort>Long</LongShort>
      <OptionType>Call</OptionType>
      <PayoffType>Asian</PayoffType>
      <ExerciseDates>
        <ExerciseDate>2020-02-15</ExerciseDate>
      </ExerciseDates>
      <Premiums> ... </Premiums>
    </OptionData>
    <Settlement>2020-02-20</Settlement>
    <ObservationDates>
        <Rules>
          <StartDate>2019-02-06</StartDate>
          <EndDate>2020-02-06</EndDate>
          <Tenor>1D</Tenor>
          <Calendar>US</Calendar>
          <Convention>F</Convention>
          <TermConvention>F</TermConvention>
          <Rule>Forward</Rule>
        </Rules>
    </ObservationDates>
  </EquityBasketOptionData>
</Trade>
```

with the following elements:

- Currency: The pay currency. Notice section "Payment Currency" in ore/Docs/ScriptedTrade.
  Allowable values: all supported currency codes, see Table 15 `Currency`.
- Notional: The quantity (for equity, commodity underlyings) / foreign amount (fx underlying)
  Allowable values: all positive real numbers
- Strike: The strike of the option
  Allowable values: all positive real numbers
- Underlyings: The basket of underlyings.
  Allowable values: for each underlying see 2.3.29
- OptionData: relevant are the long/short flag, the call/put flag, the payoff type (must be set to Asian to identify the payoff), and the exercise date (exactly one date must be given). `PayoffType2` can be optionally set to *Arithmetic* or *Geometric* and defaults to *Arithmetic* if not given. Furthermore, a *Premiums* node can be added to represent deterministic option premia to be paid by the option holder.
  Allowable values: see 2.3.1 for the general structure of the option data node
- Settlement: the settlement date (optional, if not given defaults to the exercise date)
  Allowable values: each valid date.
- ObservationDates: the observation dates for the asian
  Allowable values: See the definition in 2.3.4

The representation as a scripted trade is as follows:

```
<Trade id="AsianBasketOption#1">
  <TradeType>ScriptedTrade</TradeType>
  <Envelope/>
   <AsianBasketOptionData>
```

```
    <Expiry type="event">2020-02-15</Expiry>
    <Settlement type="event">2020-02-20</Settlement>
    <ObservationDates type="event">
      <ScheduleData>
        <Rules>
          <StartDate>2019-02-06</StartDate>
          <EndDate>2020-02-06</EndDate>
          <Tenor>1D</Tenor>
          <Calendar>US</Calendar>
          <Convention>F</Convention>
          <Rule>Forward</Rule>
        </Rules>
      </ScheduleData>
    </ObservationDates>
    <PutCall type="optionType">Call</PutCall>
    <LongShort type="longShort">Long</LongShort>
    <Notional type="number">1</Notional>
    <Strike type="number">5000</Strike>
    <Underlyings type="index">
      <Value>EQ-RIC:.SPX</Value>
      <Value>EQ-RIC:.STOXX50E</Value>
    </Underlyings>
    <Weights type="number">
      <Value>1.0</Value>
      <Value>1.0</Value>
    </Weights>
    <PayCcy type="currency">USD</PayCcy>
  </AsianBasketOptionData>
</Trade>
```

The AsianBasketOption script referenced in the trade above is shown in Listing 126.

Listing 126: Payoff script for an AsianBasketOption.

```
REQUIRE SIZE(Underlyings) == SIZE(Weights);

NUMBER d, u, basketPrice, ExerciseProbability, Payoff;
NUMBER currentNotional;

FOR d IN (1, SIZE(ObservationDates)) DO
    FOR u IN (1, SIZE(Underlyings)) DO
        basketPrice = basketPrice + Underlyings[u](ObservationDates[d]) * Weights[u];
    END;
END;

basketPrice = basketPrice / SIZE(ObservationDates);

Payoff = max(PutCall * (basketPrice - Strike), 0);

Option = LongShort * Notional * PAY(Payoff, Expiry, Settlement, PayCcy);

IF Payoff > 0 THEN
    ExerciseProbability = 1;
END;

currentNotional = Notional * Strike;
```

The meanings and allowable values of the elements in the `AsianBasketOptionData` node are given below, with data type indicated in square brackets.

- [event] `Expiry`: Option expiry date.
  Allowable values: See `Date` in Table 13.
- [event] `Settlement`: Option settlement date.
  Allowable values: See `Date` in Table 13.
- [event] `ObservationDates`: Price monitoring schedule.

Allowable values: See section 2.3.4 Schedule Data and Dates, or DerivedSchedule (see ore/Docs/ScriptedTrade).

- [optionType] `PutCall`: Option type with
  Allowable values *Call, Put*.
- [longShort] `LongShort`: Position type, *Long* if we buy, *Short* if we sell.
  Allowable values: *Long, Short*.
- [number] `Notional`: Quantity multiplier applied to the basket price
  Allowable values: Any positive real number.
- [number] `Strike`: Strike basket price in PayCcy (see below)
  Allowable values: Any positive real number.
- [index] `Underlyings`: List of underlying indices enclosed by `<Value>` and `</Value>` tags.
  See ore/Docs/ScriptedTrade's Index Section for allowable values.
- [number] `Weights`: List of weights applied to the underlying prices in the basket, given in the same order as the Underlyings above, each weight enclosed by `<Value>` and `</Value>` tags.
  Allowable values: Any positive real number.
- [currency] `PayCcy`: The payment currency. For FX, where the underlying is provided in the form `FX-SOURCE-CCY1-CCY2` (see Table 21) this should be `CCY2`. If `CCY1` or the currency of the underlying (for EQ and COMM underlyings), this will result in a quanto payoff. Notice section "Payment Currency" in ore/Docs/ScriptedTrade.
  Allowable values: See Table 15 `Currency` for allowable currency codes.

Average Strike Basket Option

The traditional trade representation is as follows, using an equity underlying in this example:

```
<Trade id="AverageStrikeBasketOption#1">
  <TradeType>EquityBasketOption</TradeType>
  <Envelope>
    <CounterParty>CPTY_A</CounterParty>
    <NettingSetId>CPTY_A</NettingSetId>
    <AdditionalFields/>
  </Envelope>
  <EquityBasketOptionData>
    <Currency>USD</Currency>
    <Notional>1</Notional>
    <Underlyings>
      <Underlying>
        <Type>Equity</Type>
        <Name>RIC:.SPX</Name>
        <Currency>USD</Currency>
        <Weight>1.0</Weight>
      </Underlying>
      <Underlying>
        <Type>Equity</Type>
        <Name>RIC:.STOXX50E</Name>
        <Weight>1.0</Weight>
      </Underlying>
    </Underlyings>
    <OptionData>
      <LongShort>Long</LongShort>
```

```
      <OptionType>Call</OptionType>
      <PayoffType>AverageStrike</PayoffType>
      <ExerciseDates>
        <ExerciseDate>2020-02-15</ExerciseDate>
      </ExerciseDates>
      <Premiums> ... </Premiums>
    </OptionData>
    <Settlement>2020-02-20</Settlement>
    <ObservationDates>
      <Rules>
        <StartDate>2019-02-06</StartDate>
        <EndDate>2020-02-06</EndDate>
        <Tenor>1D</Tenor>
        <Calendar>US</Calendar>
        <Convention>F</Convention>
        <TermConvention>F</TermConvention>
        <Rule>Forward</Rule>
      </Rules>
    </ObservationDates>
  </EquityBasketOptionData>
</Trade>
```

with the following elements:

- Currency: The pay currency. Notice section "Payment Currency" in ore/Docs/ScriptedTrade.
  Allowable values: all supported currency codes, see Table 15 `Currency`.
- Notional: The quantity (for equity, commodity underlyings) / foreign amount (fx underlying)
  Allowable values: all positive real numbers
- Underlyings: The basket of underlyings.
  Allowable values: for each underlying see 2.3.29
- OptionData: relevant are the long/short flag, the call/put flag, the payoff type (must be set to AverageStrike to identify the payoff), and the exercise date (exactly one date must be given). `PayoffType2` can be optionally set to *Arithmetic* or *Geometric* and defaults to *Arithmetic* if not given. Furthermore, a *Premiums* node can be added to represent deterministic option premia to be paid by the option holder.
  Allowable values: see 2.3.1 for the general structure of the option data node
- Settlement: the settlement date (optional, if not given defaults to the exercise date)
  Allowable values: each valid date.
- ObservationDates: the observation dates for the average strike
  Allowable values: See the definition in 2.3.4

The representation as a scripted trade is as follows:

```
<Trade id="AverageStrikeBasketOption#1">
  <TradeType>ScriptedTrade</TradeType>
  <Envelope/>
  <AverageStrikeBasketOptionData>
    <Expiry type="event">2020-02-15</Expiry>
    <Settlement type="event">2020-02-20</Settlement>
    <ObservationDates type="event">
      <ScheduleData>
        <Rules>
          <StartDate>2019-02-06</StartDate>
```

```
          <EndDate>2020-02-06</EndDate>
          <Tenor>1D</Tenor>
          <Calendar>US</Calendar>
          <Convention>F</Convention>
          <Rule>Forward</Rule>
        </Rules>
      </ScheduleData>
    </ObservationDates>
    <PutCall type="optionType">Call</PutCall>
    <LongShort type="longShort">Long</LongShort>
    <Notional type="number">1</Notional>
    <Underlyings type="index">
      <Value>EQ-RIC:.SPX</Value>
      <Value>EQ-RIC:.STOXX50E</Value>
    </Underlyings>
    <Weights type="number">
      <Value>1.0</Value>
      <Value>1.0</Value>
    </Weights>
    <PayCcy type="currency">USD</PayCcy>
  </AverageStrikeBasketOptionData>
</Trade>
```

The AverageStrikeBasketOption script referenced in the trade above is shown in Listing 127.

Listing 127: Payoff script for an AverageStrikeBasketOption.

```
REQUIRE SIZE(Underlyings) == SIZE(Weights);

NUMBER d, u, timeAverageBasketPrice, currentNotional;
FOR d IN (1, SIZE(ObservationDates)) DO
    FOR u IN (1, SIZE(Underlyings)) DO
        timeAverageBasketPrice = timeAverageBasketPrice
          + Underlyings[u](ObservationDates[d]) * Weights[u];
    END;
END;
timeAverageBasketPrice = timeAverageBasketPrice / SIZE(ObservationDates);

NUMBER expiryBasketPrice;
FOR u IN (1, SIZE(Underlyings)) DO
   expiryBasketPrice = expiryBasketPrice + Underlyings[u](Expiry) * Weights[u];
END;

NUMBER Payoff;
Payoff = max(PutCall * (expiryBasketPrice - timeAverageBasketPrice), 0);

Option = LongShort * Notional * PAY(Payoff, Expiry, Settlement, PayCcy);

NUMBER ExerciseProbability;
IF Payoff > 0 THEN
    ExerciseProbability = 1;
END;
FOR u IN (1, SIZE(Underlyings)) DO
  currentNotional = currentNotional + Notional * Underlyings[u](ObservationDates[1])
                                                * Weights[u];
END;
```

The meanings and allowable values of the elements in the `AverageStrikeBasketOptionData` node are given below, with data type indicated in square brackets.

- [event] `Expiry`: Option expiry date.
  Allowable values: See `Date` in Table 13.
- [event] `Settlement`: Option settlement date.
  Allowable values: See `Date` in Table 13.

- [event] `ObservationDates`: Price monitoring schedule.
  Allowable values: See section 2.3.4 Schedule Data and Dates, or DerivedSchedule (see ore/Docs/ScriptedTrade).
- [optionType] `PutCall`: Option type with
  Allowable values *Call, Put*.
- [longShort] `LongShort`: Position type, *Long* if we buy, *Short* if we sell.
  Allowable values: *Long, Short*.
- [number] `Notional`: Quantity multiplier applied to the basket price
  Allowable values: Any positive real number.
- [index] `Underlyings`: List of underlying indices enclosed by `<Value>` and `</Value>` tags.
  Allowable values: See ore/Docs/ScriptedTrade's Index Section for allowable values.
- [number] `Weights`: List of weights applied to the underlying prices in the basket, given in the same order as the Underlyings above, each weight enclosed by `<Value>` and `</Value>` tags.
  Allowable values: Any positive real number.
- [currency] `PayCcy`: The payment currency. For FX, where the underlying is provided in the form `FX-SOURCE-CCY1-CCY2` (see Table 21) this should be `CCY2`. If `CCY1` or the currency of the underlying (for EQ and COMM underlyings), this will result in a quanto payoff. Notice section "Payment Currency" in ore/Docs/ScriptedTrade.
  Allowable values: See Table 15 `Currency` for allowable currency codes.

Lookback Call Basket Option

The traditional trade representation is as follows, using an equity underlying in this example:

```
<Trade id="LookbackCallBasketOption#1">
  <TradeType>EquityBasketOption</TradeType>
  <Envelope>
    <CounterParty>CPTY_A</CounterParty>
    <NettingSetId>CPTY_A</NettingSetId>
    <AdditionalFields/>
  </Envelope>
  <EquityBasketOptionData>
    <Currency>USD</Currency>
    <Notional>1</Notional>
    <Underlyings>
      <Underlying>
        <Type>Equity</Type>
        <Name>RIC:.SPX</Name>
        <Currency>USD</Currency>
        <Weight>1.0</Weight>
      </Underlying>
      <Underlying>
        <Type>Equity</Type>
        <Name>RIC:.STOXX50E</Name>
        <Weight>1.0</Weight>
      </Underlying>
    </Underlyings>
    <OptionData>
      <LongShort>Long</LongShort>
      <PayoffType>LookbackCall</PayoffType>
```

```
      <ExerciseDates>
        <ExerciseDate>2020-02-15</ExerciseDate>
      </ExerciseDates>
      <Premiums> ... </Premiums>
    </OptionData>
    <Settlement>2020-02-20</Settlement>
    <ObservationDates>
      <Rules>
        <StartDate>2019-02-06</StartDate>
        <EndDate>2020-02-06</EndDate>
        <Tenor>1D</Tenor>
        <Calendar>US</Calendar>
        <Convention>F</Convention>
        <TermConvention>F</TermConvention>
        <Rule>Forward</Rule>
      </Rules>
    </ObservationDates>
  </EquityBasketOptionData>
</Trade>
```

with the following elements:

- Currency: The pay currency. Notice section "Payment Currency" in ore/Docs/ScriptedTrade.
  Allowable values: all supported currency codes, see Table 15 `Currency`.
- Notional: The quantity (for equity, commodity underlyings) / foreign amount (fx underlying)
  Allowable values: all positive real numbers
- Underlyings: The basket of underlyings.
  Allowable values: for each underlying see 2.3.29
- OptionData: relevant are the long/short flag, the payoff type (must be set to LookbackCall to identify the payoff), and the exercise date (exactly one date must be given). Furthermore, a *Premiums* node can be added to represent deterministic option premia to be paid by the option holder.
  Allowable values: see 2.3.1 for the general structure of the option data node
- Settlement: the settlement date (optional, if not given defaults to the exercise date)
  Allowable values: each valid date.
- ObservationDates: the observation dates for the lookback call
  Allowable values: See the definition in 2.3.4

The representation as a scripted trade is as follows:

```
<Trade id="LookbackCallBasketOption#1">
  <TradeType>ScriptedTrade</TradeType>
  <Envelope/>
  <LookbackCallBasketOptionData>
    <Expiry type="event">2020-02-15</Expiry>
    <Settlement type="event">2020-02-20</Settlement>
    <ObservationDates type="event">
      <ScheduleData>
        <Rules>
          <StartDate>2019-02-06</StartDate>
          <EndDate>2020-02-06</EndDate>
          <Tenor>1D</Tenor>
          <Calendar>US</Calendar>
          <Convention>F</Convention>
          <Rule>Forward</Rule>
```

```
        </Rules>
      </ScheduleData>
    </ObservationDates>
    <LongShort type="longShort">Long</LongShort>
    <Notional type="number">1</Notional>
    <Underlyings type="index">
      <Value>EQ-RIC:.SPX</Value>
      <Value>EQ-RIC:.STOXX50E</Value>
    </Underlyings>
    <Weights type="number">
      <Value>1.0</Value>
      <Value>1.0</Value>
    </Weights>
    <PayCcy type="currency">USD</PayCcy>
  </LookbackCallBasketOptionData>
</Trade>
```

The LookbackCallBasketOption script referenced in the trade above is shown in Listing 128.

Listing 128: Payoff script for a LookbackCallBasketOption.

```
REQUIRE SIZE(Underlyings) == SIZE(Weights);

NUMBER d, u, basketPrice, minBasketPrice, currentNotional;
FOR d IN (1, SIZE(ObservationDates)) DO
    basketPrice = 0;
    FOR u IN (1, SIZE(Underlyings)) DO
        basketPrice = basketPrice + Underlyings[u](ObservationDates[d]) * Weights[u];
    END;
    IF d == 1 THEN
        minBasketPrice = basketPrice;
    END;
    IF basketPrice < minBasketPrice THEN
        minBasketPrice = basketPrice;
    END;
END;

NUMBER expiryBasketPrice;
FOR u IN (1, SIZE(Underlyings)) DO
   expiryBasketPrice = expiryBasketPrice + Underlyings[u](Expiry) * Weights[u];
END;

NUMBER Payoff;
Payoff = max(expiryBasketPrice - minBasketPrice, 0);

Option = LongShort * Notional * PAY(Payoff, Expiry, Settlement, PayCcy);

NUMBER ExerciseProbability;
IF Payoff > 0 THEN
    ExerciseProbability = 1;
END;
FOR u IN (1, SIZE(Underlyings)) DO
  currentNotional = currentNotional + Notional * Underlyings[u](ObservationDates[1])
                                                * Weights[u];
END;
```

The meanings and allowable values of the elements in the `LookbackCallBasketOptionData` node are given below, with data type indicated in square brackets.

- [event] `Expiry`: Option expiry date.
  Allowable values: See `Date` in Table 13.
- [event] `Settlement`: Option settlement date.
  Allowable values: See `Date` in Table 13.

- [event] `ObservationDates`: Price monitoring schedule.
  Allowable values: See section 2.3.4 Schedule Data and Dates, or DerivedSchedule (see ore/Docs/ScriptedTrade).
- [longShort] `LongShort`: Position type, *Long* if we buy, *Short* if we sell.
  Allowable values: *Long, Short.*
- [number] `Notional`: Quantity multiplier applied to the basket price
  Allowable values: Any positive real number.
- [index] `Underlyings`: List of underlying indices enclosed by `<Value>` and `</Value>` tags.
  Allowable values: See ore/Docs/ScriptedTrade's Index Section for allowable values.
- [number] `Weights`: List of weights applied to the underlying prices in the basket, given in the same order as the Underlyings above, each weight enclosed by `<Value>` and `</Value>` tags.
  Allowable values: Any positive real number.
- [currency] `PayCcy`: The payment currency. For FX, where the underlying is provided in the form `FX-SOURCE-CCY1-CCY2` (see Table 21) this should be `CCY2`. If `CCY1` or the currency of the underlying (for EQ and COMM underlyings), this will result in a quanto payoff. Notice section "Payment Currency" in ore/Docs/ScriptedTrade.
  Allowable values: See Table 15 `Currency` for allowable currency codes.

Lookback Put Basket Option

The traditional trade representation is as follows, using an equity underlying in this example:

```
<Trade id="LookbackPutBasketOption#1">
  <TradeType>EquityBasketOption</TradeType>
  <Envelope>
    <CounterParty>CPTY_A</CounterParty>
    <NettingSetId>CPTY_A</NettingSetId>
    <AdditionalFields/>
  </Envelope>
  <EquityBasketOptionData>
    <Currency>USD</Currency>
    <Notional>1</Notional>
    <Underlyings>
      <Underlying>
        <Type>Equity</Type>
        <Name>RIC:.SPX</Name>
        <Currency>USD</Currency>
        <Weight>1.0</Weight>
      </Underlying>
      <Underlying>
        <Type>Equity</Type>
        <Name>RIC:.STOXX50E</Name>
        <Weight>1.0</Weight>
      </Underlying>
    </Underlyings>
    <OptionData>
      <LongShort>Long</LongShort>
      <PayoffType>LookbackPut</PayoffType>
      <ExerciseDates>
        <ExerciseDate>2020-02-15</ExerciseDate>
      </ExerciseDates>
```

```
      <Premiums> ... </Premiums>
    </OptionData>
    <Settlement>2020-02-20</Settlement>
    <ObservationDates>
      <Rules>
        <StartDate>2019-02-06</StartDate>
        <EndDate>2020-02-06</EndDate>
        <Tenor>1D</Tenor>
        <Calendar>US</Calendar>
        <Convention>F</Convention>
        <TermConvention>F</TermConvention>
        <Rule>Forward</Rule>
      </Rules>
    </ObservationDates>
  </EquityBasketOptionData>
</Trade>
```

with the following elements:

- Currency: The pay currency. Notice section "Payment Currency" in ore/Docs/ScriptedTrade.
  Allowable values: all supported currency codes, see Table 15 `Currency`.
- Notional: The quantity (for equity, commodity underlyings) / foreign amount (fx underlying)
  Allowable values: all positive real numbers
- Underlyings: The basket of underlyings.
  Allowable values: for each underlying see 2.3.29
- OptionData: relevant are the long/short flag, the payoff type (must be set to LookbackPut to identify the payoff), and the exercise date (exactly one date must be given). Furthermore, a *Premiums* node can be added to represent deterministic option premia to be paid by the option holder.
  Allowable values: see 2.3.1 for the general structure of the option data node
- Settlement: the settlement date (optional, if not given defaults to the exercise date)
  Allowable values: each valid date.
- ObservationDates: the observation dates for the lookback call
  Allowable values: See the definition in 2.3.4

The representation as a scripted trade is as follows:

```
<Trade id="LookbackPutBasketOption#1">
  <TradeType>ScriptedTrade</TradeType>
  <Envelope/>
   <LookbackPutBasketOptionData>
    <Expiry type="event">2020-02-15</Expiry>
    <Settlement type="event">2020-02-20</Settlement>
    <ObservationDates type="event">
      <ScheduleData>
        <Rules>
          <StartDate>2019-02-06</StartDate>
          <EndDate>2020-02-06</EndDate>
          <Tenor>1D</Tenor>
          <Calendar>US</Calendar>
          <Convention>F</Convention>
          <Rule>Forward</Rule>
        </Rules>
      </ScheduleData>
    </ObservationDates>
```

```
    <LongShort type="longShort">Long</LongShort>
    <Notional type="number">1</Notional>
    <Underlyings type="index">
      <Value>EQ-RIC:.SPX</Value>
      <Value>EQ-RIC:.STOXX50E</Value>
    </Underlyings>
    <Weights type="number">
      <Value>1.0</Value>
      <Value>1.0</Value>
    </Weights>
    <PayCcy type="currency">USD</PayCcy>
  </LookbackPutBasketOptionData>
</Trade>
```

The LookbackCallBasketOption script referenced in the trade above is shown in Listing 129.

Listing 129: Payoff script for a LookbackPutBasketOption.

```
REQUIRE SIZE(Underlyings) == SIZE(Weights);

NUMBER d, u, basketPrice, maxBasketPrice;
FOR d IN (1, SIZE(ObservationDates)) DO
    basketPrice = 0;
    FOR u IN (1, SIZE(Underlyings)) DO
        basketPrice = basketPrice + Underlyings[u](ObservationDates[d]) * Weights[u];
    END;
    IF d == 1 THEN
        maxBasketPrice = basketPrice;
    END;
    IF basketPrice > maxBasketPrice THEN
        maxBasketPrice = basketPrice;
    END;
END;

NUMBER expiryBasketPrice, Payoff;
FOR u IN (1, SIZE(Underlyings)) DO
   expiryBasketPrice = expiryBasketPrice + Underlyings[u](Expiry) * Weights[u];
END;

Payoff = max(maxBasketPrice - expiryBasketPrice, 0);
Option = LongShort * Notional * LOGPAY(Payoff, Expiry, Settlement, PayCcy);
```

The meanings and allowable values of the elements in the `LookbackPutBasketOptionData` node are given below, with data type indicated in square brackets.

- [event] `Expiry`: Option expiry date.
  Allowable values: See `Date` in Table 13.
- [event] `Settlement`: Option settlement date.
  Allowable values: See `Date` in Table 13.
- [event] `ObservationDates`: Price monitoring schedule.
  Allowable values: See section 2.3.4 Schedule Data and Dates, or DerivedSchedule (see ore/Docs/ScriptedTrade).
- [longShort] `LongShort`: Position type, *Long* if we buy, *Short* if we sell.
  Allowable values: *Long, Short.*
- [number] `Notional`: Quantity multiplier applied to the basket price
  Allowable values: Any positive real number.

- [index] `Underlyings`: List of underlying indices enclosed by `<Value>` and `</Value>` tags.
  Allowable values: See ore/Docs/ScriptedTrade's Index Section for allowable values.
- [number] `Weights`: List of weights applied to the underlying prices in the basket, given in the same order as the Underlyings above, each weight enclosed by `<Value>` and `</Value>` tags.
  Allowable values: Any positive real number.
- [currency] `PayCcy`: The payment currency. For FX, where the underlying is provided in the form `FX-SOURCE-CCY1-CCY2` (see Table 21) this should be `CCY2`. If `CCY1` or the currency of the underlying (for EQ and COMM underlyings), this will result in a quanto payoff. Notice section "Payment Currency" in ore/Docs/ScriptedTrade.
  Allowable values: See Table 15 `Currency` for allowable currency codes.

2.2.78 Worst Of Basket Swaps

Payoff

A worst of basket swap is a swap transaction written on a basket of n underlyings. The buyer receives an initial fixed amount from the seller and then pays a floating rate over the life of the contract and in turn receives coupons based on an equity notional amount, subject to a coupon trigger event.

If a knock-out event occurs for any given determination date, then the contract is deemed to have terminated at the corresponding settlement date, and all payments will cease thereafter, and no final equity amount will be paid out (or shares delivered). If no knock-out events are triggered, and a knock-in event occurs at the final determination date, then a final equity amount will be delivered to the buyer.

In the case of a physical settlement, the seller will deliver a number of shares of one of the underlying assets to the buyer at the final settlement date for the cost of an agreed strike price. For a cash settlement, the net difference between the strike price and the price at the determination date is paid, although the settlement type does not affect pricing.

We denote the determination dates as $t_1^d, \ldots, t_m^d$, its corresponding settlement dates as $t_1^s, \ldots, t_m^s$ and its corresponding fixing dates as $t_1^f, \ldots, t_m^f$. Note that t_1^d is the effective/start date of the contract, and t_1^s is when the initial fixed amount is paid.

At every settlement date t_i^s, $i = 2, \ldots, m$,

$$\begin{aligned} \text{Floating Amount} &= Q \cdot \phi \cdot \tau_i \cdot X(t_{i-1}^f) \\ \text{Coupon Amount} &= Q \cdot \delta \cdot c \cdot N \end{aligned}$$

where

- Q is the equity notional amount,
- $\phi = \pm 1$ distinguishes between long $(+1)$ and short (-1),

- $\tau_i = t_i^d - t_{i-1}^d$ is the accrual fraction for the $(i-1)^{\text{th}}$ period,
- $X(t_{i-1}^f)$ denotes the floating rate fixing for the $(i-1)^{\text{th}}$ payment,
- $\delta = 1$ if a coupon trigger event occurs on t_i^d, zero otherwise,
- c is the fixed coupon rate,
- N is the number of periods until the next coupon trigger event (for accumulating coupons). For example, if payments were not made for t_2^s and t_3^s because a coupon trigger did not occur (on t_2^d and t_3^d), but a coupon trigger did occurr at t_4^d, then $N = 3$ for t_4^s. If the coupon amounts are specified as non-accumulating, then $N = 1$ for the life of the contract,

However, if a knock-out event was triggered for (at least) one of $t_1^d, t_2^d, \ldots t_{i-1}^d$, then the Floating and Coupon amounts for $t_i^s, t_{i+1}^s, t_{i+2}^s, \ldots, t_m^s$ are all zero.

For instruments with accruing coupons, instead of a single observation (date) for a fixed coupon trigger event in a coupon period, the fixed trigger criteria is observed on a regular basis, and then the fixed coupon is scaled in proportion to the number of days in the period over which the fixed coupon trigger event was observed. For example, if the fixed trigger event occurred for 15 days out of 30 days in the coupon period, then $\delta = 0.5$.

Input

The FxWorstOfBasketSwap, EquityWorstOfBasketSwap, and CommodityWorstOfBasketSwap trade types have trade data containers (respectively):

- `FxWorstOfBasketSwapData`
- `EquityWorstOfBasketSwapData`
- `CommodityWorstOfBasketSwapData`

Listing 130 shows the structure of example trades for an Equity underlying.

Listing 130: EquityWorstOfBasketSwap data

```
<Trade id="EQ_WorstOfBasketSwap">
  <TradeType>EquityWorstOfBasketSwap</TradeType>
  <Envelope>
    ......
  </Envelope>
  <EquityWorstOfBasketSwapData>
    <LongShort>Long</LongShort>
    <Currency>EUR</Currency>
    <Quantity>1955000</Quantity>
    <Underlyings>
      <Underlying>
        .......
      </Underlying>
    </Underlyings>
    <InitialPrices>
      <InitialPrice>...</InitialPrice>
    </InitialPrices>
    <FloatingPeriodSchedule>
      <Dates>
        ...
      </Dates>
    </FloatingPeriodSchedule>
    <FixedDeterminationSchedule>
           ............
    </FixedDeterminationSchedule>
    <KnockOutDeterminationSchedule>
            ............
    </KnockOutDeterminationSchedule>
    <FloatingPayDates>
      <Dates>
        ....
      </Dates>
    </FloatingPayDates>
    <KnockInDeterminationSchedule>
           ............
    </KnockInDeterminationSchedule>
    <FixedPayDates>
      ............
    </FixedPayDates>
    <KnockOutLevels>
      <KnockOutLevel>...</KnockOutLevel>
    </KnockOutLevels>
    <FixedTriggerLevels>
      <FixedTriggerLevel>...</FixedTriggerLevel>
    </FixedTriggerLevels>
    <KnockInLevel>0.75</KnockInLevel>
    <FixedRate>0.04</FixedRate>
    <FixedAccrualSchedule>
      <Rules>
        ...
      </Rules>
    </FixedAccrualSchedule>
    <FloatingIndex>EUR-EURIBOR-3M</FloatingIndex>
    <FloatingDayCountFraction>Actual/360</FloatingDayCountFraction>
    <FloatingFixingSchedule>
          ............
    </FloatingFixingSchedule>
  </EquityWorstOfBasketSwapData>
</Trade>
```

The net cashflows under a worst of basket swap from the long perspective are:

1. Receive an initial fixed amount: `InitialFixedRate` $*$ `Quantity`,
2. At each determination date (if a knock-out has not occurred for any previous determination date):
    - Pay a floating amount: `Quantity` $* floatingRate * accrualFraction$,

- If a coupon trigger event occurs for this date, receive a fixed coupon amount (as well as any other fixed coupon amounts not previously received because a coupon trigger event had not occurred): `Quantity` $*$ `CouponRate`.

  If `AccumulatingFixedCoupons` is *True*, the coupon amounts are guaranteed to be paid out, with the amounts scaled relative to the fraction of days during the accrual period where a coupon trigger event occurs: `Quantity` $*$ `CouponRate` $*triggerAccrualFraction$

3. On the final determination date, if $worstPerformance < \min(Strike, KnockInLevel)$, pay `Quantity` $*$(`Strike`$-worstPerformance$), where $worstPerformance$ is the performance, i.e. S_T/S_0, of the worst-performing asset as of the final determination date T.

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

- `Currency`: The payout currency for all cashflows. For FX, this should be `CCY2` as defined (see Table 21) in the `Name` field in the `Underlying` node. Note that for (non-quanto) FxWorstOfBasketSwaps, this should be the domestic (`CCY2`) currency, as defined in the Name field in the `Underlying` node. For non-quanto Equity- and CommodityWorstOfBasketSwaps, this should be currency the equity or commodity is quoted in. Notice Section "Payment Currency" in ore/Docs/ScriptedTrade.
  Allowable values: See Table 15 `Currency`.

- `LongShort`: Own party position.
  Allowable values: *Long*, *Short*

- `Quantity`: The equity notional amount, or the quantity multiplier.
  Allowable values: Any number.

- `InitialFixedRate` [Optional]: The coupon rate for the initial fixed payment, to be paid to the *Long* party.
  Allowable values: Any number, as a percentage expressed in decimal form. If left blank or omitted, defaults to zero.

- `InitialFixedPayDate` [Optional]: Settlement date for the initial fixed payment.
  Allowable values: See `Date` in Table 13. If left blank or omitted, this defaults to the first date in the `FloatingPayDates` schedule.

- `Underlyings`: The basket of underlyings.
  Allowable values: For each underlying, see 2.3.29.

- `InitialPrices`: The observed price of each underlying in `Underlyings`.
  Allowable values: For each underlying, an `InitialPrice` sub-node with any positive number.

- `BermudanKnockIn` [Optional]: Whether the knock-in event is observed on a regular basis (*True*), as defined by the `KnockInDeterminationSchedule`, or only at the final date in the `FloatingPeriodSchedule` (*False*).
  Allowable values: Boolean node, allowing *Y*, *N*, *1*, *0*, *true*, *false*, etc. The full set of allowable values is given in Table 29. If left blank or omitted, defaults to *False*.

- `KnockInLevel`: The barrier level used to determine whether a knock-in event has occurred for the given knock-in observation date/s, upon which the equity amount is payable at the `KnockInPayDate`, subject to no knock-out event having occurred over the life of the contract.
  Allowable values: Any number, as a percentage of the initial prices expressed in decimal form.

- `FloatingPeriodSchedule`: Period dates used for calculating floating coupon accrual fractions, and for observing underlying prices for calculating performances.
  Allowable values: See section 2.3.4 Schedule Data and Dates/Rules, or DerivedSchedule for scripted trades (see the Event data structure in the scripted trade documentation at ore/Docs/ScriptedTrade).

- `FloatingFixingSchedule` [Optional]: Fixing dates for floating coupons linked to Ibor indices. The d^{th} date in this schedule is the fixing date for the $(d+1)^{\text{th}}$ date in the `FloatingPayDates`. For floating coupons linked to overnight indices this scheulde is not relevant, the fixing dates are derived from the FloatingPeriodScheudle and FloatingLookback.
  Allowable values: See section 2.3.4 Schedule Data and Dates/Rules, or DerivedSchedule for scripted trades (see the Event data structure in ore/Docs/ScriptedTrade). If left blank or omitted, defaults to a `DerivedSchedule` with the `FloatingPeriodSchedule` set as the base schedule.

- `FixedDeterminationSchedule` [Optional]: If `AccruingFixedCoupons` is *False*, the observation dates on which a fixed trigger event is evaluated, upon which a fixed coupon payout is made. Otherwise, there is no payout. If `AccruingFixedCoupons` is *True*, these will be the dates on which fixed coupon amounts are calculated. The coupon amount is scaled by the number of fixed trigger days relative to the total number of days in the period.
  Allowable values: See section 2.3.4 Schedule Data and Dates/Rules, or DerivedSchedule for scripted trades (see the Event data structure in ore/Docs/ScriptedTrade). If left blank or omitted, defaults to a `DerivedSchedule` with the `FloatingPeriodSchedule` set as the base schedule.

- `KnockOutDeterminationSchedule` [Optional]: Dates on which a knock-out event is evaluated, after which the instrument is considered to be terminated and no payout is made.
  Allowable values: See section 2.3.4 Schedule Data and Dates/Rules, or DerivedSchedule for scripted trades (see the Event data structure in ore/Docs/ScriptedTrade). If left blank or omitted, defaults to a `DerivedSchedule` with the `FloatingPeriodSchedule` set as the base schedule.

- `KnockInDeterminationSchedule` [Optional]: Observation dates for determining if a knock-in event has occurred. If `BermudanKnockIn` is *True*, this node is required.
  Allowable values: See section 2.3.4 Schedule Data and Dates/Rules, or DerivedSchedule for scripted trades (see the Event data structure in ore/Docs/ScriptedTrade).

- `FixedAccrualSchedule` [Optional]: The fixed coupon paid out for each

determination date in the `FixedDeterminationSchedule` is scaled according to the number of days in the `FixedAccrualSchedule` for which a fixed trigger event occurred, as a fraction of the total number of days in the period (defined by the `FixedDeterminationSchedule`). If `AccruingFixedCoupons` is *True*, this node is required.
Allowable values: See section 2.3.4 Schedule Data and Dates/Rules, or DerivedSchedule for scripted trades (see the Event data structure in ore/Docs/ScriptedTrade).

- `FloatingPayDates`: Floating coupon payment dates.
  Allowable values: See section 2.3.4 Schedule Data and Dates, Rules, or DerivedSchedule (see ore/Docs/ScriptedTrade).
- `FixedPayDates` [Optional]: Fixed coupon payment dates.
  Allowable values: See section 2.3.4 Schedule Data and Dates/Rules, or DerivedSchedule for scripted trades (see the Event data structure in ore/Docs/ScriptedTrade). If left blank or omitted, defaults to the `FloatingPayDates`.
- `KnockInPayDate` [Optional]: Settlement date for the knock-in payment, subject to a knock-in event having been triggered.
  Allowable values: See `Date` in Table 13. If left blank or omitted, this defaults to the last date in the `FloatingPayDates` schedule.
- `KnockOutLevels`: For each determination date in `KnockOutDeterminationSchedule` except for the first date, the barrier level used to determine whether a knock-out trigger event has occurred, resulting in a knock-out (i.e. contract termination).
  Allowable values: For each knock-out determination date, a `KnockOutLevel` sub-node with any positive number, as a percentage of the initial prices, expressed in decimal form.
- `FixedTriggerLevels`: For each fixed trigger determination date, the barrier level used to determine whether a coupon trigger event has occurred, upon which a fixed coupon is paid out at the corresponding payment date.
  Allowable values: For each fixed trigger evaluation date, a `FixedTriggerLevel` sub-node with any positive number, as a percentage of the initial prices expressed in decimal form.
- `FixedRate`: Rate of the fixed coupon leg.
  Allowable values: Any number, as a percentage expressed in decimal form.
- `AccumulatingFixedCoupons` [Optional]: Whether the fixed coupons are accumulating or not.
  Allowable values: Boolean node, allowing *Y*, *N*, *1*, *0*, *true*, *false*, etc. The full set of allowable values is given in Table 29. If left blank or omitted, defaults to *False*.
- `AccruingFixedCoupons` [Optional]: Whether the fixed coupons are paid out on an accrual basis (*True*), or subject to a single fixed trigger event (given by the `FixedDeterminationSchedule`).
  Allowable values: Boolean node, allowing *Y*, *N*, *1*, *0*, *true*, *false*, etc. The full set of allowable values is given in Table 29. If left blank or omitted, defaults to *False*.

- `FloatingIndex`: Underlying index of the floating leg.
  Allowable values: See ore/Docs/ScriptedTrade (section Data Node / Index).
  This must be an InterestRate underlying. Overnight indices are not supported.
- `FloatingSpread` [Optional]: Spread on the floating rate.
  Allowable values: Any number. If left blank or omitted, defaults to zero.
- `FloatingDayCountFraction`: The day count fraction for the floating leg accrual calculations.
  Allowable values: See `DayCount Convention` in Table 18.
- `Strike` [Optional]: Strike price for for each underlying in the basket. When calculating the equity amount, this is the price used to calculate the number of shares (for Equity underlyings) owing of the worst-performing asset.
  Allowable values: Any number, as a percentage of the initial prices, expressed in decimal form. If left blank or omitted, defaults to one, i.e. the initial price of the underlying.
- `FloatingLookback` [Optional]: Only applicable when the floating leg reference rate is an overnight index. The shift applied to each value date in `FloatingFixingSchedule` to obtain the fixing reference date.
  Allowable values: Any period definition in unit "Days" (e.g. *2D*, *30D*, but not *1M*). If omitted or left blank, defaults to *0D*.
- `FloatingRateCutoff` [Optional]: Only applicable when the floating leg reference rate is an overnight index. The number of fixing dates at the end of the fixing period for which the fixing value is held constant and set to the previous value.
  Allowable values: Any non-negative whole number. If left blank or omitted, defaults to zero.
- `IsAveraged` [Optional]: Only applicable when the floating leg reference rate is an overnight index (where there are multiple fixings over a period). If *True*, the average of the fixings is used to calculate the coupon. If *False*, the coupon is calculated by compounding the fixings.
  Allowable values: Boolean node, allowing *Y*, *N*, *1*, *0*, *true*, *false*, etc. The full set of allowable values is given in Table 29. Defaults to *false* If left blank or omiited.
- `IncludeSpread` [Optional]: Only applicable when the floating leg reference rate is an overnight index and if IsAveraged is false, i.e. the coupon rate is compounded. If *true* the spread is included in the compounding, otherwise it is excluded.
  Allowable values: Boolean node, allowing *Y*, *N*, *1*, *0*, *true*, *false*, etc. The full set of allowable values is given in Table 29.

Worst Of Basket Swaps can alternatively be represented as *scripted trades*, refer to ore/Docs/ScriptedTrade for an introduction.

```
<TradeType>ScriptedTrade</TradeType>
<Envelope>
   ....
</Envelope>
<WorstOfBasketSwapData>
  <LongShort type="longShort">Long</LongShort>
  <Quantity type="number">1955000</Quantity>
  <InitialFixedRate type="number">0.015</InitialFixedRate>
  <Underlyings type="index">
    <Value>EQ-RIC:.STOXX50E</Value>
```

```
    <Value>EQ-RIC:.SPX</Value>
  </Underlyings>
  <InitialPrices type="number">
    <Value>3481.44</Value>
    <Value>3714.24</Value>
  </InitialPrices>
  <DeterminationDates type="event">
    <ScheduleData>
        ....
    </ScheduleData>
  </DeterminationDates>
  <SettlementDates type="event">
    <ScheduleData>
      <Dates>
        ....
      </Dates>
    </ScheduleData>
  </SettlementDates>
  <KnockOutLevels type="number">
    <Value>...</Value>
  </KnockOutLevels>
  <CouponTriggerLevels type="number">
    <Value>...</Value>
  </CouponTriggerLevels>
  <KnockInLevel type="number">0.75</KnockInLevel>
  <CouponRate type="number">0.04</CouponRate>
  <AccumulatingCoupons type="bool">true</AccumulatingCoupons>
  <FloatingIndex type="index">EUR-EURIBOR-3M</FloatingIndex>
  <FloatingSpread type="number">0.002</FloatingSpread>
  <FloatingDayCountFraction type="dayCounter">Actual/360</FloatingDayCountFraction>
  <FixingSchedule type="event">
    <DerivedSchedule>
        ....
    </DerivedSchedule>
  </FixingSchedule>
  <Strike type="number">1.0</Strike>
  <PayCcy type="currency">EUR</PayCcy>
</WorstOfBasketSwapData>
```

The WorstOfBasketSwap script referenced in the trade above is shown in Listing 131.

The net cashflows under a (scripted) worst of basket swap from the long perspective are:

1. Receive an initial fixed amount: `InitialFixedRate` $*$ `Quantity`,
2. At each determination date (if a knock-out has not occurred for any previous determination date):
   - Pay a floating amount: `Quantity` $* floatingRate * accrualFraction$,
   - If a coupon trigger event occurs for this date, receive a fixed coupon amount (as well as any other fixed coupon amounts not previously received because a coupon trigger event had not occurred): `Quantity` $*$ `CouponRate`
3. On the final determination date, if $worstPerformance < \min(Strike, KnockInLevel)$, pay `Quantity` $*$(`Strike`$-worstPerformance$), where $worstPerformance$ is the performance, i.e. S_T/S_0, of the worst-performing asset as of the final determination date T.

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

- [longShort] `LongShort`: Own party position.
  Allowable values: *Long, Short*

- [number] `Quantity`: Quantity multiplier, or the equity notional amount.
  Allowable values: Any number.
- [number] `InitialFixedRate`: Rate of the initial fixed payment.
  Allowable values: Any number, as a percentage expressed in decimal form.
- [index] `Underlyings`: The basket of underlyings.
  Allowable values: See ore/Docs/ScriptedTrade's Index Section for allowable values.
- [number] `InitialPrices`: The agreed initial price for each underlying in the basket.
  Allowable values: Any positive number. The number of `Value` sub-nodes should match the number of underlyings. This must have the same number of entries as `Underlyings`.
- [event] `DeterminationDates`: Floating leg period dates, knock-out determination dates, and fixed trigger determination dates. The first date is only used for calculating the floating leg accrual fraction, and corresponds to the contract effective date.
  Allowable values: See Section 2.3.4 ScheduleData, or DerivedSchedule (see ore/Docs/ScriptedTrade) if derived from `SettlementDates` or `FixingSchedule`. The number of determination dates should match the number of settlement and fixing dates.
- [event] `SettlementDates`: The set of settlement dates corresponding to each determination date. While the first settlement date is unused in the calculations, this format simplifies the input, e.g. the settlement schedule can be a derived schedule, with the *DeterminationDates* as the base schedule.
  Allowable values: See section 2.3.4 ScheduleData, or DerivedSchedule (see ore/Docs/ScriptedTrade) if derived from `DeterminationDates` or `FixingSchedule`. The number of settlement dates should match the number of determination and fixing dates.
- [number] `KnockOutLevels`: For each determination date, the barrier level used to determine whether a knock-out trigger event has occurred, resulting in a knock-out (i.e. contract termination) at the corresponding settlement date.
  Allowable values: Any number, as a percentage of the initial prices expressed in decimal form. This should have the same number of entries as the number of dates in `DeterminationDates` minus one.
- [number] `CouponTriggerLevels`: For each determination date, the barrier level used to determine whether a coupon trigger event has occurred, upon which a fixed coupon is paid out at the corresponding settlement date.
  Allowable values: Any number, as a percentage of the initial prices expressed in decimal form. This should have the same number of entries as the number of dates in `DeterminationDates` minus one.
- [number] `KnockInLevel`: For the final determination date, the barrier level used to determine whether a knock-in event has occurred, upon which the equity amount is payable at the final settlement date, subject to no knock-out event having occurred in the life of the contract.

Allowable values: Any number, as a percentage of the initial price expressed in decimal form.

- [number] `CouponRate`: Rate of the fixed coupon leg.
  Allowable values: Any number, as a percentage expressed in decimal form.
- [bool] `AccumulatingCoupons`: Whether the fixed coupons are accumulating or not.
  Allowable values: Boolean node, allowing *Y*, *N*, *1*, *0*, *true*, *false*, etc. The full set of allowable values is given in Table 29.
- [index] `FloatingIndex`: Underlying index of the floating leg.
  Allowable values: See ore/Docs/ScriptedTrade's Index Section for allowable values. Overnight indices are not supported.
- [number] `FloatingSpread`: Spread on the floating rate.
  Allowable values: Any number.
- [dayCounter] `FloatingDayCountFraction`: The day count fraction for the floating leg accrual calculations.
  Allowable values: See `DayCount Convention` in Table 18.
- [event] `FixingSchedule`: The fixing schedule of the floating leg payments.
  Allowable values: See Section 2.3.4 ScheduleData, or DerivedSchedule (see ore/Docs/ScriptedTrade) if derived from `DeterminationDates` or `SettlementDates`. The number of fixing dates should match the number of determination and settlement dates.
- [number] `Strike`: Strike price for each underlying in the basket. When calculating the equity amount, this is the price used to calculate the number of shares owing of the worst-performing asset.
  Allowable values: Any number, as a percentage of the initial prices, expressed in decimal form.
- [currency] `PayCcy`: Settlement currency. Notice section "Payment Currency" in ore/Docs/ScriptedTrade.
  Allowable values: See Table 15 `Currency`.

2.2.79 Accumulators and Decumulators

Payoff

An Equity accumulator is an exotic product that requires the investor (or provides the option) to buy shares of an underlying security at a pre-determined strike price K. The product settles periodically (e.g. monthly) and allows the investor to accumulate a holding in the underlying security over the term of the contract.

A potential feature of an accumulator is a knock-out, i.e. the contract terminates early if the underlying price exceeds some threshold above strike.

Another feature is leverage, i.e. the investor buys a certain number of shares per day when the security trades below K, and an enhanced (leveraged) number of shares when the security trades at or above K.

The accumulator investor speculates that the security trades between strike price K and the knock-out level thoughout the life of the contract.

A decumulator is the reverse of an accumulator – in this case, the investor is required to sell shares periodically at a pre-determined fixed price. A potential knock-out is set below the strike price (and below the spot price at inception).

Accumulators/decumulators are also traded on FX underlyings, Commodity underlyings are possible as well.

Input

The `FxAccumulatorData`, `EquityAccumulatorData`, `CommodityAccumulatorData` is the trade data container for the FxAccumulator, EquityAccumulator, CommodityAccumulator trade type. The following listings and show the structure of example trades for an FX and Equity underlying. Here the FX accumulator is of "type 01" meaning that a settlement takes place on each observation date while the equity accumulator is of "type 02" meaning that a settlement takes place on specific period end dates for all observation dates in that period.

```
<Trade id="FX_ACCUMULATOR">
  <TradeType>FxAccumulator</TradeType>
  <Envelope>
   ...
  </Envelope>
  <FxAccumulatorData>
    <Currency>USD</Currency>
    <FixingAmount>1000000</FixingAmount>
    <Strike>1.1</Strike>
    <Underlying>
      <Type>FX</Type>
      <Name>ECB-EUR-USD</Name>
    </Underlying>
    <OptionData>
      <LongShort>Long</LongShort>
      <PayoffType>Accumulator</PayoffType>
    </OptionData>
    <StartDate>2016-03-01</StartDate>
    <ObservationDates>
      <Dates>
        <Dates>
          <Date>2017-03-01</Date>
          <Date>2020-03-01</Date>
          <Date>2025-03-01</Date>
          <Date>2029-03-01</Date>
        </Dates>
      </Dates>
    </ObservationDates>
    <SettlementDates>
      <Dates>
        <Dates>
          <Date>2017-03-03</Date>
          <Date>2020-03-03</Date>
          <Date>2025-03-03</Date>
          <Date>2029-03-03</Date>
        </Dates>
     </Dates>
```

```
    </SettlementDates>
    <RangeBounds>
      <RangeBound>
        <RangeTo>1.14</RangeTo>
        <Leverage>1</Leverage>
      </RangeBound>
      <RangeBound>
        <RangeFrom>1.14</RangeFrom>
        <Leverage>1</Leverage>
      </RangeBound>
    </RangeBounds>
    <Barriers>
      <BarrierData>
        <Type>UpAndOut</Type>
        <Style>American</Style>
        <Levels>
          <Level>1.5</Level>
        </Levels>
      </BarrierData>
      <BarrierData>
        <Type>FixingFloor</Type>
        <Levels>
          <Level>2</Level>
        </Levels>
      </BarrierData>
    </Barriers>
  </FxAccumulatorData>
</Trade>
```

```
<Trade id="Equity_Decumulator">
  <TradeType>EquityAccumulator</TradeType>
  <Envelope>
   ...
  </Envelope>
  <EquityAccumulatorData>
    <FixingAmount>30</FixingAmount>
    <StrikeData>
      <Value>4000</Value>
      <Currency>EUR</Currency>
    </StrikeData>
    <Underlying>
        <Type>Equity</Type>
        <Name>.STOXX50</Name>
        <IdentifierType>RIC</IdentifierType>
    </Underlying>
    <OptionData>
      <LongShort>Long</LongShort>
      <PayoffType>Decumulator</PayoffType>
    </OptionData>
    <StartDate>20190925</StartDate>
    <ObservationDates>
      <Rules>
        <StartDate>20190925</StartDate>
        <EndDate>20200925</EndDate>
        <Tenor>1D</Tenor>
        <Calendar>TARGET</Calendar>
        <Convention>F</Convention>
```

```
        <TermConvention>F</TermConvention>
        <Rule>Forward</Rule>
      </Rules>
    </ObservationDates>
    <PricingDates>
      <Dates>
        <Dates>
          <Date>20211025</Date>
          <Date>20211125</Date>
          ...
        </Dates>
      </Dates>
    </PricingDates>
    <SettlementLag>2D</SettlementLag>
    <SettlementCalendar>TARGET</SettlementCalendar>
    <SettlementConvention>F</SettlementConvention>
    <RangeBounds>
      <RangeBound>
        <RangeTo>4000</RangeTo>
        <Leverage>1</Leverage>
      </RangeBound>
      <RangeBound>
        <RangeFrom>4000</RangeFrom>
        <Leverage>2</Leverage>
      </RangeBound>
    </RangeBounds>
    <Barriers>
      <BarrierData>
        <Type>DownAndOut</Type>
        <LevelData>
          <Level>
            <Value>3500</Value>
            <Currency>EUR</Currency>
          </Level>
        </LevelData>
      </BarrierData>
      <BarrierData>
        <Type>FixingFloor</Type>
        <Levels>
          <Level>1</Level>
        </Levels>
      </BarrierData>
    </Barriers>
    <KnockOutSettlementAtPeriodEnd>false<KnockOutSettlementAtPeriodEnd>
  </EquityAccumulatorData>
</Trade>
```

`Accumulator Payout Formula`

The payout formula, from the perspective of the party that is long, for each observation date of an Accumulator is:

$$Payout = RangeBound(Leverage) \cdot FixingAmount \cdot (fix - Strike),$$

i.e. for an Accumulator the holder pays [`Strike` * `FixingAmount` * `RangeBound`(Leverage)] expressed in `Currency` and receives/buys [fix * `FixingAmount`

* RangeBound(Leverage)]'s worth of equity/CCY1/commodity in Currency on each observation date.

If NakedOption is *true*, the payout formula for an Accumulator (long perspective) is:

$$Payout = |RangeBound(Leverage)| \cdot FixingAmount \cdot \max(0, \phi \cdot (fix - Strike)),$$

where $\phi = \pm 1$ is the option type (+1 for Call, -1 for Put) inferred from the sign of the values of RangeBound(Leverage), which are all required to be the same.

Decumulator Payout Formula

The payout formula, from the perspective of the party that is long, for each observation date of a Decumulator is:

$$Payout = RangeBound(Leverage) \cdot FixingAmount \cdot (Strike - fix),$$

For a Decumulator the holder pays/sells [fix * FixingAmount * RangeBound(Leverage)]'s worth of equity/CCY1/commodity in Currency, and receives [Strike * FixingAmount * RangeBound(Leverage)] expressed in Currency on each observation date.

If NakedOption is *true*, the payout formula for a Decumulator (long perspective) is:

$$Payout = |RangeBound(Leverage)| \cdot FixingAmount \cdot \max(0, \phi \cdot (Strike - fix),$$

where $\phi = \pm 1$ is the option type (+1 for Call, -1 for Put) inferred from the sign of the values of RangeBound(Leverage), which are all required to be the same.

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

- StrikeData [Optional]: A node containing the global strike in Value and the currency in which both the underlying and the strike are quoted in Currency. Only supported for EquityAccumulators.

  Allowable values: See Currency in Table 13. The strike may be any positive real number. The currency provided in this node may be quoted as corresponding minor currency to the underlying major currency. If omitted, local strikes should be used in each RangeBound node.

- Currency: The payout currency. The result of the payout formula above is treated to be in this currency. Note that for (non-quanto) FxAccumulators this should be the domestic (CCY2) currency. For non-quanto Equity- and CommodityAccumulators this should be the currency the equity or commodity is quoted in. Notice section "Payment Currency" in ore/Docs/ScriptedTrade. Allowable values: See Table 15 Currency.

- FixingAmount: The unleveraged notional amount accumulated at each fixing date.

For FxAccumulators: The FixingAmount is expressed in the foreign currency (`CCY1`).

For EquityAccumulators: The FixingAmount is expressed as number of shares/units of the underlying equity or equity index.

For CommodityAccumulators: The FixingAmount is expressed as number of units of the underlying commodity.

Allowable values: Any real number. Note that a negative amount causes an Accumulator to become a Decumulator, and vice-versa.

- DailyFixingAmount [Optional]: For accumulator type "01" only: If *true*, the fixing amount for a period is calculated as the given FixingAmount times the number of calendar days in the period. The periods are given by [StartDate, ObservationDate(1)], [ObservationDate(1), ObservationDate(2)], ... etc. i.e. if *true*, a StartDate is required to determine the calendar days in the first period. If *false*, the fixing amount is used directly without any weighting.

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

- Strike [Optional]: Global strike associated to the ranges. Is overwritten by local strikes defined in `RangeBounds` or modified by range strike adjustments. Note that each RangeBound needs a strike, either the global strike defined here, or a local one in the `RangeBounds` nodes which then overwrites the global one.

  For FxAccumulators: The the fx strike rate (global and local) is defined as amount in domestic currency (`CCY2`) for one unit of foreign currency (`CCY1`).

  For Equity- and CommodityAccumulators: The strike value (global and local) for one unit/share/contract of the underlying equity or commodity, expressed in the currency the equity or commodity is quoted or in any other pay currency. In case of a composite option style a FxIndex is required (see below). For EquityAccumulators, the `StrikeData` node (see above) should be used.

  The logic for global and local strikes is as follows:

  - if a local Strike for an interval is given, this local Strike will be used for the interval (a global Strike will be ignored if given)
  - otherwise if a global Strike and a local StrikeAdjustment are given, the strike used for the interval will be global Strike + local StrikeAdjustment
  - otherwise if a global Strike, but no local StrikeAdjustment is given, the strike used for the interval will be the global Strike
  - if no global strike and no local strike is given for an interval, an error is thrown, since the strike information is not sufficient

  Allowable values: Any positive real number.

- Underlying: A node with the underlying of the Accumulator instrument.

  For FxAccumulators: `Type` is set to *FX* and `Name` is a string of the form *SOURCE-CCY1-CCY2* where `CCY1` is the foreign currency, `CCY2` is the domestic currency, and *SOURCE* is the fixing source, see Table 21 and 2.3.29.

For EquityAccumulators: `Type` is set to *Equity* and `Name` and other fields are as outlined in 2.3.29

For CommodityAccumulators: `Type` is set to *Commodity* and `Name` is an identifier of the commodity as outlined in 2.3.29 and in Table 25.

Allowable values: Any FX, Equity or Commodity underlying as specified in 2.3.29

- OptionData: See 2.3.1. A node that describes whether the instrument is *Long* or *Short*, whether the payoff type is *Accumulator* or *Decumulator*,

  The relevant fields in the `OptionData` node for an Accumulator are:

  – `LongShort` The allowable values are *Long* or *Short*. Note that a *Long Accumulator* is equivalent to a *Short Decumulator* except when there are guaranteed fixings, i.e. when a Barrier node of Type *FixingFloor* is present. The *FixingFloor* causes asymmetry in the payoff.

  `PayoffType` The allowable values are *Accumulator* or *Decumulator*. For a long accumulator the strike is paid and the underlying received, for a long decumulator it is the other way around.

- StartDate [Mandatory for American Style Barrier or if a daily fixing amount is used, Optional for European Style or no Barrier and no daily fixing amount is used]: Used to set the start of the knock out monitoring - American knock out events are monitored from this date on. This field is only relevant for accumulators of type American Knock-Out, and not used otherwise. Can be omitted for European knock outs, or if there is no barrier.
  Allowable values: See `Date` in Table 13.

- ObservationDates: The observation dates given as schedule data.
  Allowable values: See the ScheduleData definition 2.3.4.

- PricingDates[Optional]: The dates defining observation period end dates on which the settlement for all dates in the period takes place. Note that the inclusion or not of PricingDates determines the Accumulator type. If included the Accumulator is of "type 02", and otherwise, if PricingDates are not included it is of "type 01".
  Allowable values: See the ScheduleData definition 2.3.4. Note that the final pricing period end date (defining the final observation period end date) must be on or after the final observation date.

- SettlementLag [Optional]: The settlement delay. Optional, if not given it is defaulted to 0D.
  Allowable values: Any period definition (e.g. *2D*, *1W*, *1M*, *1Y*)

- SettlementCalendar [Optional]: The calendar used to compute the settlement date from the corresponding observation date.
  Allowable values: Any valid calendar, see Table 17 Calendar.. Optional, defaults to the null calendar (no holidays).

- SettlementConvention [Optional]: The convention used to compute the settlement date from the corresponding observation date. Optional, defaults to F (following).

Allowable values: Any valid roll convention (*U,F,P,MF,MP*), see Table 14 Roll Convention.

- SettlementDates [Optional]: The settlement dates can also be given as an explicit list of dates, see the FX accumulator listing above for an example. For a "type 01" accumulator the number of dates must be equal to the number of observation dates. For a "type 02" accumulator the number of dates must be equal to the number of pricing dates. If an explicit list of settlement dates is given, no settlement lag, calendar, conventions should be given.

- NakedOption [Optional]: If *true*, the payoff represents that of an option, and only positive values are accumulated in the instrument. The option type (Call or Put) is inferred from the sign of the `Leverage` values in `RangeBound`, which are all required to be the same.
  Allowable values: Boolean node, allowing *Y*, *N*, *1*, *0*, *true*, *false*, etc. The full set of allowable values is given in Table 29. Defaults to *false* if left blank or omitted.

- RangeBounds: Nodes for the specification of ranges and associated leverages. A leverage can be specified to apply for all values the underlying can take, or for specific ranges using `RangeFrom` and `RangeTo`. Multiple `RangeBound` sub-nodes can be included within the `RangeBounds` node. If no leverage is specified for a range, it defaults to 1 for this range. In addition a range bound specific strike can be specified for Accumulators of "type 01" (and only this type!) which overwrites the global Strike field. If all range bounds have a specific strike defined, the global Strike field might be omitted.
  Allowable values: For each range, see 2.3.32. Only the `Leverage` is relevant for a given range. All `Leverage` parameters in one instrument must have the same sign. For "type 01" Accumulators, the Strike is relevant too and overwrites the global strike if given. Finally, for "type 01" Accumulators a range bound specific strike can be specified by specifying a range bound specific strike adjustment to the global strike (see the range bound spec for details).

- Barriers [Optional]: Specification of barriers and fixing floors (guaranteed fixings). Multiple `BarrierData` sub-nodes can be included within the `Barriers` node. Relevant fields for each `BarrierData` sub-node are `Type`, `Style`, and `Level`. The barrier is monitored on the

  – the observation dates (type 02 accumulator, i.e. PricingDates are given)

  – the observation dates (type 01 accumulator, i.e. no PricingDates are given), if barrier style is European

  – from the start date to the first observation date and between the observation dates on a continuous basis (type 01 accumulator), if barrier style is American

  For *type 01 accumulators* (no pricing dates are given), the FixingFloor guarantees a specific number of fixings to be settled even in case of a knock out. On a guaranteed fixing date, Only positive payouts (from the buyer/long perspective) are realised. Where payout = fix - strike for accumulators and strike - fix for decumulators.

  For *type 02 accumulators* (pricing dates are given), the FixingFloor specifies the

number of *periods* (rather than observation dates) that are guaranteed to settle. If a knock out event occurs within a settlement period the fixings after the knock event within this period are assumed to fall into the first defined range and the settlement takes place on the knock out day plus the defined settlement delay.

The barrier `Level` for `Type` *UpAndOut* and *DownAndOut* is expressed in the same way as `Strike` outlined above. For EquityAccumulators, each `Level` node should be provided a `Value` and a `Currency` and contained in a `LevelData` node (see example trade above). This `Currency` supports minor currencies.
The barrier `Level` for `Type` *FixingFloor* is a non-negative integer.

Allowable values: For each `BarrierData` node, see 2.3.31. For Accumulators/Decumulators, the following values for `Type` are relevant: *UpAndOut*, *DownAndOut* and *FixingFloor*. The barrier `Style` can be *European* (monitored on observation dates) or *American* (continuously monitored).

- KnockOutSettlementAtPeriodEnd [Optional]: Only relevant for *type 02 accumulator*. Controlls the settlement behavior in case of an knock out events. If *true* the settlement of a knock out event takes place at the end of the observation period, otherwise on the knock out event date plus the defined settlement delay. Defaults to *false*.
- KnockOutFixingAtKOSettlement [Optional]: Only relevant for *type 02 accumulator*. Controlls the settlement behavior in case of an knock out events. If *true* the fixing at knockout settlement date is used to compute the knockout flow, otherwise the fixing at knockout date. Defaults to *false*.
- FxIndex [Optional]: Required for composite accumulators of type 2, where the strike, knockout barrier and pay currency are different from the underlying quote currency. Intended for Commodity and Equity Accumulators. A node with the underlying of the FX conversion from quote currency to pay currency. `Type` is set to *FX* and `Name` is a string of the form *SOURCE-CCY1-CCY2* where `CCY1` is the foreign (quote) currency, `CCY2` is the domestic (pay) currency, and *SOURCE* is the fixing source, see Table 21 and 2.3.29.

Accumulators can also be represented as scripted trades, refer to the separate documentation in ore/Docs/ScriptedTrade for an introduction. Listing 132 shows the structure of an Accumulator (type 01) example, here on a FX underlying (EQ or COMM underlyings are possible as well).

Listing 132: Accumulator (type 01) Scripted Representation

```
<Trade id="SCRIPTED_FX_ACCUMULATOR">
  <TradeType>ScriptedTrade</TradeType>
  <Envelope>
    <CounterParty>CPTY_A</CounterParty>
    <NettingSetId>CPTY_A</NettingSetId>
    <AdditionalFields/>
  </Envelope>
  <Accumulator01Data>
    <Strike type="number">1.1</Strike>
    <FixingAmount type="number">1000000</FixingAmount>
    <LongShort type="longShort">Long</LongShort>
    <Underlying type="index">FX-ECB-EUR-USD</Underlying>
    <PayCcy type="currency">USD</PayCcy>
    <StartDate type="event">2016-03-01</StartDate>
    <FixingDates type="event">
      <ScheduleData>
        <Dates>
          <Dates>
            <Date>2017-03-01</Date>
            <Date>2020-03-01</Date>
            <Date>2025-03-01</Date>
            <Date>2029-03-01</Date>
          </Dates>
        </Dates>
      </ScheduleData>
    </FixingDates>
    <SettlementDates type="event">
      <DerivedSchedule>
        <BaseSchedule>FixingDates</BaseSchedule>
        <Shift>2D</Shift>
        <Calendar>TARGET</Calendar>
        <Convention>F</Convention>
      </DerivedSchedule>
    </SettlementDates>
    <RangeUpperBounds type="number">
      <Value>1.14</Value>
      <Value>10000</Value>
    </RangeUpperBounds>
    <RangeLowerBounds type="number">
      <Value>0</Value>
      <Value>1.14</Value>
    </RangeLowerBounds>
    <RangeLeverages type="number">
      <Value>1</Value>
      <Value>2</Value>
    </RangeLeverages>
    <KnockOutLevel type="number">1.5</KnockOutLevel>
    <KnockOutType type="barrierType">UpOut</KnockOutType>
    <AmericanKO type="bool">true</AmericanKO>
    <GuaranteedFixings type="number">2</GuaranteedFixings>
  </Accumulator01Data>
</Trade>
```

The meanings and allowable values of the elements in the `Accumulator01Data` representation follow below.

- Strike: The strike value the bought currency is purchased at.
- FixedAmount: The unleveraged notional amount accumulated at each fixing date
- LongShort: 1 for a long, -1 for a short position
- Underlying: See ore/Docs/ScriptedTrade's Index section for allowable values.
- PayCcy: The payment currency of the trade. Notice section Notice section "Payment Currency" in ore/Docs/ScriptedTrade.
- StartDate: The start date. American knock out events are monitored from this date on. Notice that the start date must be given in the scripted trade representation for European knock outs, although it is not used for this variant.
- FixingDates: The fixing dates, given as a ScheduleData, or DerivedSchedule (see ore/Docs/ScriptedTrade).
- SettlementDates: The fixing dates, given as a ScheduleData, or DerivedSchedule (see ore/Docs/ScriptedTrade).
- RangeUpperBound: Values of upperbounds for the leverage ranges. If a given range has no upperbound add 100000
- RangeLowerBound: Values of lowerbounds for the leverage ranges. If a given range has no lowerbound add 0
- RangeLeverages: Values of leverages for the leverage ranges.
- KnockOutLevel: The KnockOut Barrier level
- KnockOutType: The KnockOut Barrier type, can be UpOut or DownOut
- AmericanKO: If true, knock out events are monitored on a continuous basis, otherwise they are monitored on the fixing dates only.
- GuaranteedFixings: Number of the first n Fixings that are guaranteed, regardless of whether or not the trade has been knocked out.

The script 'Accumulator01' referenced in the trade above is shown in Listing 133.

Listing 133: Accumulator type 01 script

```
REQUIRE KnockOutType == 3 OR KnockOutType == 4;
NUMBER Payoff, fix, d, r, Alive, currentNotional, Factor, ThisPayout, Fixing[SIZE(FixingDates)];
Alive = 1;
FOR d IN (1, SIZE(FixingDates), 1) DO
    fix = Underlying(FixingDates[d]);
    Fixing[d] = fix;

    IF AmericanKO == 1 THEN
      IF KnockOutType == 4 THEN
        IF FixingDates[d] >= StartDate THEN
           IF d == 1 OR FixingDates[d-1] <= StartDate THEN
              Alive = Alive * (1 - ABOVEPROB(Underlying, StartDate, FixingDates[d], KnockOutLevel));
           ELSE
              Alive = Alive * (1 - ABOVEPROB(Underlying, FixingDates[d-1], FixingDates[d], KnockOutLevel));
           END;
        END;
      ELSE
        IF FixingDates[d] >= StartDate THEN
           IF d == 1 OR FixingDates[d-1] <= StartDate THEN
              Alive = Alive * (1 - BELOWPROB(Underlying, StartDate, FixingDates[d], KnockOutLevel));
           ELSE
              Alive = Alive * (1 - BELOWPROB(Underlying, FixingDates[d-1], FixingDates[d], KnockOutLevel));
           END;
        END;
      END;
    ELSE
      IF {KnockOutType == 4 AND fix >= KnockOutLevel} OR
         {KnockOutType == 3 AND fix <= KnockOutLevel} THEN
        Alive = 0;
      END;
    END;

    IF d <= GuaranteedFixings THEN
      Factor = 1;
    ELSE
      Factor = Alive;
    END;

    FOR r IN (1, SIZE(RangeUpperBounds), 1) DO
      IF fix > RangeLowerBounds[r] AND fix <= RangeUpperBounds[r] THEN
        ThisPayout = RangeLeverages[r] * FixingAmount * (fix - Strike) * Factor;
        IF d > GuaranteedFixings OR ThisPayout >= 0 THEN
          Payoff = Payoff + LOGPAY(RangeLeverages[r] * FixingAmount * (fix - Strike) * Factor,
                                   FixingDates[d], SettlementDates[d], PayCcy);
        END;
      END;
    END;
END;
value = LongShort * Payoff;
currentNotional = FixingAmount * Strike;
```

Accumulators of Type 02 can also be represented as scripted trades, see ore/Docs/ScriptedTrade for an introduction. Listing 134 shows the structure of an Accumulator (Type 02) example, here on an equity underlying. FX and COMM underlyings are possible as well.

Listing 134: EQ Accumulator (type 02) Scripted Representation

```
<Trade id="EqDecumulator">
  <TradeType>ScriptedTrade</TradeType>
  <Envelope>
    <CounterParty>CPTY_A</CounterParty>
    <NettingSetId>CPTY_A</NettingSetId>
    <AdditionalFields/>
  </Envelope>
  <Accumulator02Data>
    <Strike type="number">59.9842</Strike>
    <FixingAmount type="number">11</FixingAmount>
    <LongShort type="longShort">Long</LongShort>
    <Underlying type="index">EQ-RIC:.SPX</Underlying>
    <PayCcy type="currency">EUR</PayCcy>
    <ObservationDates type="event">
      <ScheduleData>
        <Rules>
          <StartDate>20190925</StartDate>
          <EndDate>20200925</EndDate>
          <Tenor>1D</Tenor>
          ...
        </Rules>
      </ScheduleData>
    </ObservationDates>
    <KnockOutSettlementDates type="event">
      <DerivedSchedule>
        <BaseSchedule>ObservationDates</BaseSchedule>
        <Shift>2D</Shift>
        <Calendar>TARGET</Calendar>
        <Convention>F</Convention>
      </DerivedSchedule>
    </KnockOutSettlementDates>
    <ObservationPeriodEndDates type="event">
      <ScheduleData>
        <Dates>
          <Dates>
            <Date>20191025</Date>
            <Date>20191125</Date>
            ...
            <Date>20200925</Date>
          </Dates>
        </Dates>
      </ScheduleData>
    </ObservationPeriodEndDates>
    <SettlementDates type="event">
      <ScheduleData>
        <Dates>
          <Dates>
            <Date>20191027</Date>
            <Date>20191127</Date>
            ...
            <Date>20200927</Date>
          </Dates>
        </Dates>
      </ScheduleData>
    </SettlementDates>
    <RangeUpperBounds type="number">
      <Value>59.9842</Value>
      <Value>10000000.0</Value>
    </RangeUpperBounds>
    <RangeLowerBounds type="number">
      <Value>-10000000.0</Value>
      <Value>59.9842</Value>
```

The script 'Accumulator02' referenced in the trade above is shown in Listing 135.

The meanings and allowable values of the elements in the `DecumulatorData` node folllow below.

- Strike: The forward price. For an FX underlying `FX-SOURCE-CCY1-CCY2` this is the number of units of `CCY2` per units of `CCY1`. For an EQ underlying this is the equity price expressed in the equity ccy. Allowable values are non-negative numbers.
- FixingAmount: The number of shares per day (EQ Undlerinyg) resp. the foreign amount (FX Undderlying). Allowable values are non-negative values.
- LongShort: The position, allowable values are "Long" and "Short"
- Underlying: The underlying index
  See ore/Docs/ScriptedTrade's Index section for allowable values.
- PayCcy: The payment currency. See the appendix for allowable currency codes. Notice section Notice section "Payment Currency" in ore/Docs/ScriptedTrade.
- ObservationDates: The observation date schedule. See ore/Docs/ScriptedTrade on how this is set up.
- KnowOutSettlementDates: The settlement dates associated to the observation dates in case of a knock out event, the number of observation and knock out settlement dates must be equal. See ore/Docs/ScriptedTrade on how this is set up.
- ObservationPeriodEndDates: The last date for each observation period. See ore/Docs/ScriptedTrade on how this is set up.
- SettlementDates: The settlement dates for each observation period, the number of settlement dates and the number of observation period end dates must be equal. See ore/Docs/ScriptedTrade on how this is set up.
- RangeUpperBounds: The multiplier for the "number of days below" in the payoff. Allowable values are non-negative numbers.
- RangeLowerBounds: The multiplier for the "number of days above" in the payoff. Allowable values are non-negative numbers.
- RangeLeverages: The multiplier for the defined ranges. Allowable values are non-negative numbers.
- DefaultRange: The default range used in case of a knock-out event to classify the remaining days until the GuaranteedPeriodEndDate. Allowable values are $1, 2, \ldots, r$ where r is the number of defined ranges.
- KnockOutLevel: The knock out price. See Strike for more details and allowable values.
- KnockOutType: The KnockOut Barrier type, can be UpOut or DownOut
- GuaranteedPeriodEndDate: The last date of the guaranteed period. Allowable values are any valid dates.

Listing 135: Accumulator type 02 script.

```
REQUIRE SIZE(ObservationDates) == SIZE(KnockOutSettlementDates);
REQUIRE SIZE(ObservationPeriodEndDates) == SIZE(SettlementDates);
REQUIRE SIZE(RangeUpperBounds) == SIZE(RangeLowerBounds);
REQUIRE SIZE(RangeUpperBounds) == SIZE(RangeLeverages);
NUMBER Payoff, fix, d, dd, KnockedOut, currentNotional, Days[SIZE(RangeUpperBounds)];
NUMBER currentPeriod, r;
currentPeriod = 1;
FOR d IN (1, SIZE(ObservationDates)) DO
  fix = Underlying(ObservationDates[d]);
  IF KnockedOut == 0 THEN
    IF {KnockOutType == 4 AND fix >= KnockOutLevel} OR
       {KnockOutType == 3 AND fix <= KnockOutLevel} THEN
       KnockedOut = 1;
       FOR dd IN (d + 1, SIZE(ObservationDates)) DO
         IF ObservationDates[d] <= GuaranteedPeriodEndDate THEN
            Days[DefaultRange] = Days[DefaultRange] + 1;
         END;
       END;
       FOR r IN (1, SIZE(RangeUpperBounds)) DO
         value = value + PAY( LongShort * FixingAmount * RangeLeverages[r] * Days[r]
                              * ( fix - Strike ),
                              ObservationDates[d], KnockOutSettlementDates[d], PayCcy );
       END;
    END;
  END;
  IF KnockedOut == 0 THEN
    FOR r IN (1, SIZE(RangeUpperBounds)) DO
      IF fix > RangeLowerBounds[r] AND fix <= RangeUpperBounds[r] THEN
        Days[r] = Days[r] + 1;
      END;
    END;
    IF ObservationDates[d] >= ObservationPeriodEndDates[currentPeriod] THEN
      FOR r IN (1, SIZE(RangeUpperBounds)) DO
        value = value + PAY( LongShort * FixingAmount * RangeLeverages[r] * Days[r]
                             * ( fix - Strike ),
                             ObservationDates[d], SettlementDates[currentPeriod], PayCcy );
      END;
    END;
  END;
  IF ObservationDates[d] >= ObservationPeriodEndDates[currentPeriod] THEN
    currentPeriod = currentPeriod + 1;
    IF currentNotional == 0 THEN
      FOR r IN (1, SIZE(RangeUpperBounds)) DO
        currentNotional = currentNotional + FixingAmount * RangeLeverages[r] * Days[r] * Strike;
      END;
    END;
    FOR r IN (1, SIZE(RangeUpperBounds)) DO
      Days[r] = 0;
    END;
  END;
END;
```

2.2.80 Target Redemption Forward (TaRF)

Payoff

A TaRF is a sequence of (FX or Equity or Commodity) forward contracts, specified by some strike(s) K_k and leverage factors L_k applicable to ranges $[A_k, B_k]$, $k = 1, \ldots, K$, for the underlying price X_i at a sequence of expiry dates T_i. If a T_i lies in the future, the associated fixing X_i is not known at the valuation date.

The trade accumulates positive profits

$$P_i = P_{i-1} + \sum_{k=1}^{K} N\, L_k\, \max(X_i - K_k, 0)\, 1_{A_k < X_i \leq B_k}, \qquad i \geq 1, \qquad P_0 = 0$$

and terminates (knocks out) under some conditions on P_i:

There are two types of knock out

- A Continuous TaRF terminates when the profit reaches or exceeds a pre-determined target.
- A Digital TaRF terminates when the number of fixing days where the buyer makes a profit reaches a pre-determined target.

The payment after the knock out event can be of different types

- exact: the payment is adjusted to match the pre-determined target
- full: the unadjusted payment is made
- truncated: no payment is made

Additional features include

- European Knock Ins (EKI), where no sells or buys happen if a fixing is between the strike and the EKI level
- Pivots, where different strikes apply for different ranges of the fixing

Input

The `FxTaRFData`, `EquityTaRFData`, `CommodityTaRFData` is the trade data container for the FxTaRF, EquityTaRF, CommodityTaRF trade type, listings 136 and 137 show the structure of example trades for an FX and Equity underlying.

Listing 136: FxTaRF data

```
<Trade id="FX_TARF">
  <TradeType>FxTaRF</TradeType>
  <Envelope/>
  <FxTaRFData>
    <Currency>USD</Currency>
    <FixingAmount>1000000</FixingAmount>
    <TargetAmount>100000</TargetAmount>
    <Strike>1.1</Strike>
    <Underlying>
      <Type>FX</Type>
      <Name>ECB-EUR-USD</Name>
    </Underlying>
    <ScheduleData>
      <Dates>
        <Dates>
          <Date>2017-03-01</Date>
          <Date>2020-03-01</Date>
          <Date>2025-03-01</Date>
          <Date>2029-03-01</Date>
        </Dates>
      </Dates>
    </ScheduleData>
    <OptionData>
      <LongShort>Long</LongShort>
      <PayoffType>TargetExact</PayoffType>
    </OptionData>
    <RangeBounds>
      <RangeBound>
        <RangeTo>1.05</RangeTo>
        <Leverage>2</Leverage>
      </RangeBound>
      <RangeBound>
        <RangeFrom>1.1</RangeFrom>
        <Leverage>1</Leverage>
      </RangeBound>
    </RangeBounds>
    <Barriers>
      <BarrierData>
        <Type>CumulatedProfitCap</Type>
        <Levels>
          <Level>100000</Level>
        </Levels>
      </BarrierData>
    </Barriers>
  </FxTaRFData>
</Trade>
```

Listing 137: EquityTaRF data

```
<Trade id="EQ_TARF">
  <TradeType>EquityTaRF</TradeType>
  <Envelope/>
  <EquityTaRFData>
    <Currency>USD</Currency>
    <FixingAmount>775</FixingAmount>
    <Strike>2900</Strike>
    <Underlying>
      <Type>Equity</Type>
      <Name>RIC:.SPX</Name>
    </Underlying>
    <ScheduleData>
      <Dates>
        <Dates>
          <Date>2017-03-01</Date>
          <Date>2020-03-01</Date>
          <Date>2025-03-01</Date>
          <Date>2026-03-01</Date>
          <Date>2027-03-01</Date>
          <Date>2028-03-01</Date>
        </Dates>
      </Dates>
    </ScheduleData>
    <OptionData>
      <LongShort>Long</LongShort>
      <PayoffType>TargetFull</PayoffType>
    </OptionData>
    <RangeBounds>
        <RangeBound>
            <RangeTo>3300</RangeTo>
            <Leverage>1</Leverage>
        </RangeBound>
        <RangeBound>
            <RangeFrom>3300</RangeFrom>
            <Leverage>2</Leverage>
            <StrikeAdjustment>100</StrikeAdjustment>
        </RangeBound>
    </RangeBounds>
    <Barriers>
      <BarrierData>
        <Type>FixingCap</Type>
        <Levels>
          <Level>3</Level>
        </Levels>
      </BarrierData>
    </Barriers>
  </EquityTaRFData>
</Trade>
```

The payout formula for each fixing date of a TaRF - for a Long position - is:
$Payout = RangeBound(Leverage) \cdot FixingAmount \cdot (fix - Strike(global/local))$

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

- Currency: The payout currency of the TaRF. The TargetAmount and the result

of the payout formula above are expressed in this currency. For FX, this should be `CCY2` as defined in the Name field in the Underlying node. Note that for (non-quanto) FxTaRFs this should be the domestic (`CCY2`) currency, as defined in the Name field in the Underlying node. For non-quanto Equity- and CommodityTaRFs this should be the currency the equity or commodity is quoted in. Notice section "Payment Currency" in ore/Docs/ScriptedTrade.
Allowable values: See Table 15 `Currency`.

- FixingAmount: The unlevered FixingAmount amount to be used at each fixing date, as per the payout formula above.

  For FxTaRFs: The FixingAmount is expressed in the foreign currency (`CCY1`).

  For EquityTaRFs: The FixingAmount is expressed as number of shares/units of the underlying equity or equity index.

  For CommodityTaRFs: The FixingAmount is expressed as number of units of the underlying commodity.

  Allowable values: Any positive real number. Note that the FixingAmount must be positive both for Long and Short positions.

- TargetAmount[Optional]: If PayoffType = *TargetExact*, after a knock-out event the last payout is adjusted so that the cumulated profit since the start of the TaRF matches the target amount. TargetAmount is optional, only required for PayoffType = *TargetExact*. The TargetAmount is a currency amount, for FxTaRFs it is in (`CCY2`).

  Notice that if the barrier type is *CumulatedProfitCapPoints* the last payout is adjusted in a way such that the cumulated profit in relative terms (as defined for this barrier type below) matches the target amount divided by the fixing amount (i.e. the target amount in points).
  Allowable values: Any positive real number.

- TargetPoints[Optional]: For PayoffType = *TargetExact*, this provides an alternative way of expressing the TargetAmount as a percentage of FixingAmount in decimal form. For example a FixingAmount = $1,000,000$ and TargetPoints = 0.0700 translates to a TargetAmount = $70,000$.
  Allowable values: Any positive real number.

- Strikes [Optional]: Global strikes associated to the ranges. Either this node or the Strike node must be given to specify (a) global strike(s) valid across all ranges. If no such global strikes are given, the strikes have to be specified as part of the range bound definition. The Strikes node represents time-varying strikes while the Strike node represents a strike that is constant over time. If the Strikes node is given, it must contain a number of Strike subnodes. Either

  – the Strike subnodes except the first one must contain an attribute startDate which indicates from which fixing date on the strike is valid. The first strike is valid until the minimum startDate given in the other nodes, or

  – the values in the Strike subnodes are associated to the given fixing dates in their order, the number of values must be equal to the number of fixing

dates or less, in the latter case the last given value is repeated to match the number of fixing dates.

See the description under Strike for further information on the single strike values and the relation to strike information given in range bounds.

- Strike [Optional]: Global strike associated to the ranges. Is overwritten by local strikes defined in `RangeBounds` or modified by range strike adjustments. Note that each RangeBound needs a strike, either the global strike defined here, or a local one in the `RangeBounds` nodes which then overwrites the global one. For FX, the strike is expressed as amount in `CCY2` per one unit of `CCY1` as defined in the Name field in the Underlying node.

  The logic for global and local strikes is as follows:

  - if a local Strike for a range is given, this local Strike will be used for the range (a global Strike will be ignored if given)
  - otherwise if a global Strike and a local StrikeAdjustment are given, the strike used for the range will be global Strike + local StrikeAdjustment
  - otherwise if a global Strike, but no local StrikeAdjustment is given, the strike used for the range will be the global Strike
  - if no global strike and no local strike is given for an range, an error is thrown, since the strike information is not sufficient

  Allowable values: Any positive real number.

- Underlying: A node with the underlying of the TaRF instrument.

  For FxTaRFs: `Type` is set to *FX* and `Name` is a string on the form *SOURCE-CCY1-CCY2* where `CCY1` is the foreign currency, `CCY2` is the domestic currency, and *SOURCE* is the fixing source, see Table 21 and 2.3.29.

  For EquityTaRFs: `Type` is set to *Equity* and `Name` and other fields are as outlined in 2.3.29

  For CommodityTaRFs: `Type` is set to *Commodity* and `Name` is an identifier of the commodity as outlined in 2.3.29 and in Table 25.

  Allowable values: Any FX, Equity or Commodity underlying as specified in 2.3.29

- ScheduleData: The fixing dates schedule of the TaRF.
  Allowable values: See 2.3.4

- SettlementLag [Optional]: The settlement delay.
  Allowable values: Any period definition (e.g. *2D*, *1W*, *1M*, *1Y*). If omitted or left blank, defaults to *0D*.

- SettlementCalendar [Optional]: The calendar used to compute the settlement date from the corresponding fixing date.
  Allowable values: Any valid calendar, see Table 17 Calendar. If omitted or left blank, defaults to the *NullCalendar* (no holidays).

- SettlementConvention [Optional]: The convention used to compute the settlement date from the corresponding fixing date.

Allowable values: Any valid roll convention (*U,F,P,MF,MP*), see Table 14 Roll Convention. If omitted or left blank, defaults to *F* (following).

- OptionData: This is a trade component sub-node outlined in section 2.3.1. The relevant fields in the `OptionData` node for an TaRF are:
  - `LongShort` The allowable values are *Long* or *Short*.
  - `PayoffType` can be *TargetFull*, *TargetExact*, or *TargetTruncated*, impacting the payment after a Knock-Out event as per:

    TargetExact: the payment is set to the pre-determined `TargetAmount`
    TargetFull: the unadjusted payment is made
    TargetTruncated: no payment is made

- RangeBoundSet: Time-varying range-bound information. Either this node or the RangeBounds node (for time-independent range bounds) must be given. If the RangeBoundSet node is given, it must contain a number of RangeBounds nodes. Either
  - the RangeBounds subnodes except the first one must contain an attribute startDate which indicates from which fixing date on the values are valid. The first value is valid until the minimum startDate given in the other nodes, or
  - the values in the RangeBounds subnodes are associated to the given fixing dates in their order, the number of RangeBounds subnodes must be equal to the number of fixing dates or less, in the latter case the last given RangeBounds subnode is repeated to match the number of fixing dates.

  See the description under RangeBounds for further information on the single RangeBounds nodes.

- RangeBounds: Nodes for the specification of ranges (intervals) and associated `Leverage`, local `Strike` and local `StrikeAdjustment`. A range/interval can be specified to apply for all values the underlying can take, or for specific ranges using `RangeFrom` and `RangeTo`. Multiple `RangeBound` sub-nodes can be included within the `RangeBounds` node. If the value for `RangeFrom` is omitted it is interpreted as $-\infty$ (unbounded towards minus infinity). If `RangeTo` is omitted it is interpreted as ∞ (unbounded towards plus infinity). Note that if the underlying takes a value not covered by any range/interval, no payout will occur.

  For FxTaRFs: The the `Strike`, `StrikeAdjustment`, `RangeFrom` and `RangeTo` Fx rates are all defined as amount in domestic currency (`CCY2`) for one unit of foreign currency (`CCY1`).

  For Equity- and CommodityTaRFs: The `Strike`, `StrikeAdjustment`, `RangeFrom` and `RangeTo` are all defined as the value of one unit/share/contract of the underlying equity or commodity, expressed in the currency the equity or commodity is quoted in.
  Allowable values: For each range, see 2.3.32. Note that an interval/range can't have both a local `Strike` and a local `StrikeAdjustment`. If no global `Strike` is given, each interval/range must have a local `Strike`.

- Barriers: Specification of barriers, see 2.3.31. Multiple `BarrierData` sub-nodes can be included within the `Barriers` node. Relevant fields for each TaRF `BarrierData` sub-node are `Type` and `Level`. For a TaRF, `Type` can be set to *CumulatedProfitCap*, *CumulatedProfitCapPoints* or *FixingCap*. Notice that *CumulatedProfitCapPoints* can not be combined with the other barrier types.
  - *CumulatedProfitCap*: The TaRF terminates once the generated profit from the long side's view reaches the CumulatedProfitCap. (Continuous TaRF) For an FxTaRF the CumulatedProfitCap is quoted in `CCY2`.
    Allowable values: The `Level` for a *CumulatedProfitCap* can be set to any non-negative real number.
  - *CumulatedProfitCapPoints*: The TaRF terminates once the generated profit from the long side's view reaches the CumulatedProfitCapPoints. (Continuous TaRF) Here, the generated profit is measured as the absolute profit amount (as used in CumulatedProfitCap) divided by the FixingAmount and divided by the absolute value of the leverage.
    Allowable values: The `Level` for a *CumulatedProfitCapPoints* can be set to any non-negative real number.
  - *FixingCap*: The TaRF terminates once number of fixings where a profit is generated - from the long side's view -reaches the FixingCap. (Digital TaRF)
    Allowable values: The `Level` for a *FixingCap* can be set to a non-negative integer.

Target Redemption Forwards can alternatively represented as *scripted trades*, refer to the scripted trade documentation in ore/Docs/ScriptedTrade for an introduction. This representation does not allow for time varying strikes or range bounds and also not for a target amount / cumulated profit cap specification in points.

```
<Trade id="TaRF#1">
  <TradeType>ScriptedTrade</TradeType>
  <Envelope/>
  <TaRFData>
    <FixingAmount type="number">1000000</FixingAmount>
    <LongShort type="longShort">Long</LongShort>
    <Underlying type="index">FX-ECB-EUR-USD</Underlying>
    <PayCcy type="currency">USD</PayCcy>
    <FixingDates type="event">
      <ScheduleData>
        <Dates>
          <Dates>
              <Date>2017-03-01</Date>
              <Date>2020-03-01</Date>
              <Date>2025-03-01</Date>
              <Date>2029-03-01</Date>
          </Dates>
        </Dates>
      </ScheduleData>
    </FixingDates>
    <SettlementDates type="event">
      <DerivedSchedule>
        <BaseSchedule>FixingDates</BaseSchedule>
```

```
        <Shift>2D</Shift>
        <Calendar>TARGET</Calendar>
        <Convention>F</Convention>
      </DerivedSchedule>
    </SettlementDates>
    <RangeUpperBounds type="number">
        <Value>1.05</Value>
        <Value>10000</Value>
    </RangeUpperBounds>
    <RangeLowerBounds type="number">
        <Value>0</Value>
        <Value>1.1</Value>
    </RangeLowerBounds>
    <RangeLeverages type="number">
        <Value>2</Value>
        <Value>1</Value>
    </RangeLeverages>
    <RangeStrikes type="number">
      <Value>1.1</Value>
      <Value>1.1</Value>
    </RangeStrikes>
    <KnockOutProfitAmount type="number">100000</KnockOutProfitAmount>
    <KnockOutProfitEvents type="number">0</KnockOutProfitEvents>
    <TargetAmount type="number">100000</TargetAmount>
    <TargetType type="number">0</TargetType>
  </TaRFData>
</Trade>
```

The TaRF script referenced in the trade above is shown in Listing 138.

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

- [number] `FixingAmount`: Amount to buy/sell on each fixing day should be FixingAmount times Leverage of the range that the spot level fixes within.
  Allowable values: Any positive real number.
- [longShort] `LongShort`: *Long* if we buy the option. *Short* if we sell the option.
  Allowable values: *Long, Short*
- [index] `Underlying`: Underlying index.
  Allowable values: See ore/Docs/ScriptedTrade's Index Section for allowable values.
- [currency] `PayCcy`: The payment currency. For FX, where the underlying is provided in the form `FX-SOURCE-CCY1-CCY2` (see Table 21) this should be `CCY2`. If `CCY1` or the currency of the underlying (for EQ and COMM underlyings), this will result in a quanto payoff. Notice section "Payment Currency" in ore/Docs/ScriptedTrade.
  Allowable values: See Table 15 for allowable currency codes.
- [event] `FixingDates`: The fixing dates, given as a ScheduleData node
  Allowable values: See section 2.3.4 Schedule Data and Dates, or DerivedSchedule (see the scripted trade documentation in ore/Docs/ScriptedTrade).
- [event] `SettlementDates`: The settlement dates, given as a ScheduleData or DerivedSchedule node. Allowable values: See section 2.3.4 Schedule Data and

Dates, or DerivedSchedule (see the scripted trade documentation in ore/Docs/ScriptedTrade).

- [number] `RangeUpperBound`: Values of upperbounds for the leverage ranges. If a given range has no upperbound, add 100000.
  Allowable values: Any list of positive real numbers. *RangeUpperBound*, *RangeLowerBound* and *RangeLeverages* must have the same size. $RangeUpperBound[i] \geq RangeLowerBound[i]$ for all i
- [number] `RangeLowerBound`: Values of lowerbounds for the leverage ranges. If a given range has no lowerbound add 0
  Allowable values: Any list of positive real numbers.
- [number] `RangeLeverages`: Values of leverages for the leverage ranges. Positive for Bullish TaRF, Negative for Bearish TaRF.
  Allowable values: Any list of real numbers.
- [number] `RangeStrikes`: Strikes associated to the ranges. .
  Allowable values: Any positive real number.
- [number] `KnockOutProfitAmount`: If positive, cap of the profit the buyer can make.
  Allowable values: Any positive real number or zero if not applicable.
- [number] `KnockOutProfitEvents`: If positive, cap of the number of fixing days on which buyer makes profit.
  Allowable values: Any positive real number or zero if not applicable.
- [number] `TargetAmount`: The target amount to be paid as an accumulated profit for TargetType = exact.
  Allowable values: Any positive real number or zero if not applicable.
- [number] `TargetType`: *-1* for truncated, *0* for exact, *1* for full. If DigitalKnockOut is true the target type must be set to full.
  Allowable values: *-1,0,1*

2.2.81 Knock Out Swap

A Knock Out Swap refers to a vanilla fixed vs. float Interest Rate Swap that terminates when the float index fixing is above ("up and out") or below ("down and out") a given barrier level.

A Knock Out Swap is represented using the TradeType *KnockOutSwap* and a KnockOutSwapData block as shown in listing 139. It must have one `BarrierData` node, and two legs, one fixed and one floating, each represented by a `LegData` trade component.

A Knock Out Swap is a Swap with one Fixed and one Floating leg, where the Swap is terminated if the Floating leg Index hits a barrier. The barrier is monitored on all floating leg fixing dates after the BarrierStartDate.

The meanings and allowable values in this block are as follows:

- BarrierData: This node specifies the barrier Type and Level. The Type must be *UpAndOut* or *DownAndOut*. Exactly one Level must be specified. All other

values of the barrier data block are not relevant.
Allowable values: See 2.3.31.

- BarrierStartDate: The barrier is monitored on all floating leg fixing dates that are on or after the barrier start date.
Allowable values: See `Date` in Table 13.

- LegData: This specifies the swap terms. Exactly two LegData nodes must be given, one of type Fixed and one of type Floating.
Allowable values: See 2.3.3.

2.2.82 Rainbow Options

Rainbow options are European calls or puts on the maximum or mimimum of of a range of assets. We denote the prices of n assets at expiry $S_1, S_2, \ldots, S_n$ and initial prices $S_1^0, S_2^2, \ldots, S_n^0$ with respective constant weights w_1,...,w_n. The payoff at expiry of the rainbow options covered here is

Rainbow Options are represented as traditional trades or *scripted trades*, refer to the scripted trade documentation in ore/Docs/ScriptedTrade for an introduction of the latter. Each of the supported variations, all European style, is represented by a separate payoff script as shown in Table 5 (excluding Rainbow Call Spread Options and Worst Performance Rainbow Options).

Rainbow Option Payoff	Payoff Script Name
$\max(w_1 S_1, \ldots, w_n S_n, K)$	BestOfAssetOrCashRainbowOption
$\min(w_1 S_1, \ldots, w_n S_n, K)$	WorstOfAssetOrCashRainbowOption
$\max(\omega(\max(w_1 S_1, \ldots, w_n S_n) - K, 0)$	MaxRainbowOption
$\max(\omega(\min(w_1 S_1, \ldots, w_n S_n) - K, 0)$	MinRainbowOption

Table 5: Rainbow option types and associated script names. $\omega = \pm 1$ distinguishes call (+1) and put (-1).

The supported underlying types are Equity, Fx or Commodity resulting in corresponding trade types and trade data container names

- EquityRainbowOption / EquityRainbowOptionData
- FxRainbowOption / FxRainbowOptionData
- CommodityRainbowOption / CommodityRainbowOptionData

Trade input and the associated payoff script are described in the following for 12 supported Rainbow Option variations.

Best Of Asset Or Cash Rainbow Option

The traditional trade representation is as follows, using an equity underlying in this example:

```
<Trade id="BestOfAssetOrCashRainbowOption#1">
  <TradeType>EquityRainbowOption</TradeType>
  <Envelope>
    <CounterParty>CPTY_A</CounterParty>
```

```
    <NettingSetId>CPTY_A</NettingSetId>
    <AdditionalFields/>
  </Envelope>
  <EquityRainbowOptionData>
    <Currency>USD</Currency>
    <Notional>1</Notional>
    <Strike>2000</Strike>
    <Underlyings>
      <Underlying>
        <Type>Equity</Type>
        <Name>RIC:.SPX</Name>
        <Weight>1.0</Weight>
      </Underlying>
      <Underlying>
        <Type>Equity</Type>
        <Name>RIC:.STOXX50E</Name>
        <Currency>EUR</Currency>
        <Weight>1.0</Weight>
      </Underlying>
    </Underlyings>
    <OptionData>
      <LongShort>Long</LongShort>
      <PayoffType>BestOfAssetOrCash</PayoffType>
      <ExerciseDates>
        <ExerciseDate>2020-02-15</ExerciseDate>
      </ExerciseDates>
    </OptionData>
    <Settlement>2020-02-20</Settlement>
  </EquityRainbowOptionData>
</Trade>
```

with the following elements:

- Currency: The pay currency. Notice section "Payment Currency" in ore/Docs/ScriptedTrade.
  Allowable values: all supported currency codes, see Table 15 `Currency`.
- Notional: The quantity (for equity, commodity underlyings) / foreign amount (fx underlying)
  Allowable values: all positive real numbers
- Strike: The strike of the option
  Allowable values: all positive real numbers
- Underlyings: The basket of underlyings.
  Allowable values: for each underlying see 2.3.29
- OptionData: relevant are the long/short flag, the payoff type (must be set to BestOfAssetOrCash to identify the payoff), and the exercise date (exactly one date must be given)
  Allowable values: see 2.3.1 for the general structure of the option data node
- Settlement: the settlement date (optional, if not given defaults to the exercise date)
  Allowable values: each valid date.

The representation as a scripted trade is as follows:

```
<Trade id="BestOfAssetOrCashRainbowOption#1">
  <TradeType>ScriptedTrade</TradeType>
  <Envelope/>
  <BestOfAssetOrCashRainbowOptionData>
    <Expiry type="event">2020-02-15</Expiry>
    <Settlement type="event">2020-02-20</Settlement>
```

```
    <LongShort type="longShort">Long</LongShort>
    <Notional type="number">1</Notional>
    <Strike type="number">2000</Strike>
    <Underlyings type="index">
      <Value>EQ-RIC:.SPX</Value>
      <Value>EQ-RIC:.STOXX50E</Value>
    </Underlyings>
    <Weights type="number">
      <Value>1.0</Value>
      <Value>1.0</Value>
    </Weights>
    <PayCcy type="currency">USD</PayCcy>
  </BestOfAssetOrCashRainbowOptionData>
</Trade>
```

The BestOfAssetOrCashRainbowOption script referenced in the trade above is shown in Listing 140.

The meanings and allowable values of the elements in the `BestOfAssetOrCashRainbowOptionData` node are given below, with data type indicated in square brackets.

- [event] `Expiry`: Option expiry date.
  Allowable values: See `Date` in Table 13.
- [event] `Settlement`: Option settlement date.
  Allowable values: See `Date` in Table 13.
- [longShort] `LongShort`: Position type, *Long* if we buy, *Short* if we sell.
  Allowable values: *Long, Short.*
- [number] `Notional`: Quantity multiplier applied to the basket price
  Allowable values: Any positive real number.
- [number] `Strike`: Strike price in PayCcy (see below)
  Allowable values: Any positive real number.
- [index] `Underlyings`: List of underlying indices enclosed by `<Value>` and `</Value>` tags.
  Allowable values: See ore/Docs/ScriptedTrade's Index Section for allowable values.
- [number] `Weights`: List of weights applied to each of the underlying prices, given in the same order as the Underlyings above, each weighted enclosed by `<Value>` and `</Value>` tags.
  Allowable values: Any positive real number.
- [currency] `PayCcy`: The payment currency. For FX, where the underlying is provided in the form `FX-SOURCE-CCY1-CCY2` (see Table 21) this should be `CCY2`. If `CCY1` or the currency of the underlying (for EQ and COMM underlyings), this will result in a quanto payoff. Notice section "Payment Currency" in ore/Docs/ScriptedTrade.
  Allowable values: See Table 15 for allowable currency codes.

Worst Of Asset Or Cash Rainbow Option

The traditional trade representation is as follows, using an equity underlying in this example:

```
<Trade id="WorstOfAssetOrCashRainbowOption#1">
  <TradeType>EquityRainbowOption</TradeType>
  <Envelope>
    <CounterParty>CPTY_A</CounterParty>
    <NettingSetId>CPTY_A</NettingSetId>
    <AdditionalFields/>
  </Envelope>
  <EquityRainbowOptionData>
    <Currency>USD</Currency>
    <Notional>1</Notional>
    <Strike>2000</Strike>
    <Underlyings>
      <Underlying>
        <Type>Equity</Type>
        <Name>RIC:.SPX</Name>
        <Currency>USD</Currency>
        <Weight>1.0</Weight>
      </Underlying>
      <Underlying>
        <Type>Equity</Type>
        <Name>RIC:.STOXX50E</Name>
        <Weight>1.0</Weight>
      </Underlying>
    </Underlyings>
    <OptionData>
      <LongShort>Long</LongShort>
      <PayoffType>WorstOfAssetOrCash</PayoffType>
      <ExerciseDates>
        <ExerciseDate>2020-02-15</ExerciseDate>
      </ExerciseDates>
    </OptionData>
    <Settlement>2020-02-20</Settlement>
  </EquityRainbowOptionData>
</Trade>
```

with the following elements:

- Currency: The pay currency. Notice section "Payment Currency" in ore/Docs/ScriptedTrade.
  Allowable values: all supported currency codes, see Table 15 `Currency`.
- Notional: The quantity (for equity, commodity underlyings) / foreign amount (fx underlying)
  Allowable values: all positive real numbers
- Strike: The strike of the option
  Allowable values: all positive real numbers
- Underlyings: The basket of underlyings.
  Allowable values: for each underlying see 2.3.29
- OptionData: relevant are the long/short flag, the payoff type (must be set to WorstOfAssetOrCash to identify the payoff), and the exercise date (exactly one date must be given)
  Allowable values: see 2.3.1 for the general structure of the option data node
- Settlement: the settlement date (optional, if not given defaults to the exercise date)
  Allowable values: each valid date.

The representation as a scripted trade is as follows:

```
<Trade id="WorstOfAssetOrCashRainbowOption#1">
```

```
  <TradeType>ScriptedTrade</TradeType>
  <Envelope/>
  <WorstOfAssetOrCashRainbowOptionData>
    <Expiry type="event">2020-02-15</Expiry>
    <Settlement type="event">2020-02-20</Settlement>
    <LongShort type="longShort">Long</LongShort>
    <Notional type="number">1</Notional>
    <Strike type="number">2000</Strike>
    <Underlyings type="index">
      <Value>EQ-RIC:.SPX</Value>
      <Value>EQ-RIC:.STOXX50E</Value>
    </Underlyings>
    <Weights type="number">
      <Value>1.0</Value>
      <Value>1.0</Value>
    </Weights>
    <PayCcy type="currency">USD</PayCcy>
  </WorstOfAssetOrCashRainbowOptionData>
</Trade>
```

The WorstOfAssetOrCashRainbowOption script referenced in the trade above is shown in Listing 141.

The meanings and allowable values of the elements in the `BestOfAssetOrCashRainbowOptionData` node are given below, with data type indicated in square brackets.

- [event] `Expiry`: Option expiry date.
  Allowable values: See `Date` in Table 13.
- [event] `Settlement`: Option settlement date.
  Allowable values: See `Date` in Table 13.
- [longShort] `LongShort`: Position type, *Long* if we buy, *Short* if we sell.
  Allowable values: *Long, Short.*
- [number] `Notional`: Quantity multiplier applied to the basket price
  Allowable values: Any positive real number.
- [number] `Strike`: Strike price in PayCcy (see below)
  Allowable values: Any positive real number.
- [index] `Underlyings`: List of underlying indices enclosed by `<Value>` and `</Value>` tags.
  Allowable values: See ore/Docs/ScriptedTrade's Index Section for allowable values.
- [number] `Weights`: List of weights applied to each of the underlying prices, given in the same order as the Underlyings above, each weighted enclosed by `<Value>` and `</Value>` tags.
  Allowable values: Any positive real number.
- [currency] `PayCcy`: The payment currency. For FX, where the underlying is provided in the form `FX-SOURCE-CCY1-CCY2` (see Table 21) this should be `CCY2`. If `CCY1` or the currency of the underlying (for EQ and COMM underlyings), this will result in a quanto payoff. Notice section "Payment Currency" in ore/Docs/ScriptedTrade.
  Allowable values: See Table 15 for allowable currency codes.

Put/Call on Max Rainbow Option

The traditional trade representation is as follows, using an equity underlying in this example:

```
<Trade id="MaxRainbowOption#1">
  <TradeType>EquityRainbowOption</TradeType>
  <Envelope>
    <CounterParty>CPTY_A</CounterParty>
    <NettingSetId>CPTY_A</NettingSetId>
    <AdditionalFields/>
  </Envelope>
  <EquityRainbowOptionData>
    <Currency>USD</Currency>
    <Notional>1</Notional>
    <Strike>3000</Strike>
    <Underlyings>
      <Underlying>
        <Type>Equity</Type>
        <Name>RIC:.SPX</Name>
        <Weight>1.0</Weight>
      </Underlying>
      <Underlying>
        <Type>Equity</Type>
        <Name>RIC:.STOXX50E</Name>
        <Weight>1.0</Weight>
      </Underlying>
    </Underlyings>
    <OptionData>
      <LongShort>Long</LongShort>
      <OptionType>Call</OptionType>
      <PayoffType>MaxRainbow</PayoffType>
      <ExerciseDates>
        <ExerciseDate>2020-02-15</ExerciseDate>
      </ExerciseDates>
    </OptionData>
    <Settlement>2020-02-20</Settlement>
  </EquityRainbowOptionData>
</Trade>
```

with the following elements:

- Currency: The pay currency. Notice section "Payment Currency" in ore/Docs/ScriptedTrade.
  Allowable values: all supported currency codes, see Table 15 `Currency`.
- Notional: The quantity (for equity, commodity underlyings) / foreign amount (fx underlying)
  Allowable values: all positive real numbers
- Strike: The strike of the option
  Allowable values: all positive real numbers
- Underlyings: The basket of underlyings.
  Allowable values: for each underlying see 2.3.29
- OptionData: relevant are the long/short flag, the option type, the payoff type (must be set to MaxRainbow to identify the payoff), and the exercise date (exactly one date must be given)
  Allowable values: see 2.3.1 for the general structure of the option data node
- Settlement: the settlement date (optional, if not given defaults to the exercise date)

Allowable values: each valid date.

The representation as a scripted trade is as follows:

```
<Trade id="MaxRainbowOption#1">
  <TradeType>ScriptedTrade</TradeType>
  <Envelope/>
  <MaxRainbowOptionData>
    <Expiry type="event">2020-02-15</Expiry>
    <Settlement type="event">2020-02-20</Settlement>
    <PutCall type="optionType">Call</PutCall>
    <LongShort type="longShort">Long</LongShort>
    <Notional type="number">1</Notional>
    <Strike type="number">3000</Strike>
    <Underlyings type="index">
      <Value>EQ-RIC:.SPX</Value>
      <Value>EQ-RIC:.STOXX50E</Value>
    </Underlyings>
    <Weights type="number">
      <Value>1.0</Value>
      <Value>1.0</Value>
    </Weights>
    <PayCcy type="currency">USD</PayCcy>
  </MaxRainbowOptionData>
</Trade>
```

The MainRainbowOption script referenced in the trade above is shown in Listing 142.

The meanings and allowable values of the elements in the `MaxRainbowOptionData` node are given below, with data type indicated in square brackets.

- [event] `Expiry`: Option expiry date.
  Allowable values: See `Date` in Table 13.
- [event] `Settlement`: Option settlement date.
  Allowable values: See `Date` in Table 13.
- [optionType] `PutCall`: Option type with
  Allowable values *Call, Put.*
- [longShort] `LongShort`: Position type, *Long* if we buy, *Short* if we sell.
  Allowable values: *Long, Short.*
- [number] `Notional`: Quantity multiplier applied to the basket price
  Allowable values: Any positive real number.
- [number] `Strike`: Strike price in PayCcy (see below)
  Allowable values: Any positive real number.
- [index] `Underlyings`: List of underlying indices enclosed by `<Value>` and `</Value>` tags.
  Allowable values: See ore/Docs/ScriptedTrade's Index Section for allowable values.
- [number] `Weights`: List of weights applied to each of the underlying prices, given in the same order as the Underlyings above, each weight enclosed by `<Value>` and `</Value>` tags.
  Allowable values: Any positive real number.
- [currency] `PayCcy`: The payment currency. For FX, where the underlying is provided in the form `FX-SOURCE-CCY1-CCY2` (see Table 21) this should be `CCY2`.

If CCY1 or the currency of the underlying (for EQ and COMM underlyings), this will result in a quanto payoff. Notice section "Payment Currency" in ore/Docs/ScriptedTrade.
Allowable values: See Table 15 for allowable currency codes.

Put/Call on Min Rainbow Option

The traditional trade representation is as follows, using an equity underlying in this example:

```
<Trade id="MinRainbowOption#1">
  <TradeType>EquityRainbowOption</TradeType>
  <Envelope>
    <CounterParty>CPTY_A</CounterParty>
    <NettingSetId>CPTY_A</NettingSetId>
    <AdditionalFields/>
  </Envelope>
  <EquityRainbowOptionData>
    <Currency>USD</Currency>
    <Notional>1</Notional>
    <Strike>2000</Strike>
    <Underlyings>
      <Underlying>
        <Type>Equity</Type>
        <Name>RIC:.SPX</Name>
        <Weight>1.0</Weight>
      </Underlying>
      <Underlying>
        <Type>Equity</Type>
        <Name>RIC:.STOXX50E</Name>
        <Weight>1.0</Weight>
      </Underlying>
    </Underlyings>
    <OptionData>
      <LongShort>Long</LongShort>
      <OptionType>Call</OptionType>
      <PayoffType>MinRainbow</PayoffType>
      <ExerciseDates>
        <ExerciseDate>2020-02-15</ExerciseDate>
      </ExerciseDates>
    </OptionData>
    <Settlement>2020-02-20</Settlement>
  </EquityRainbowOptionData>
</Trade>
```

with the following elements:

- Currency: The pay currency. Notice section "Payment Currency" in ore/Docs/ScriptedTrade.
  Allowable values: all supported currency codes, see Table 15 `Currency`.
- Notional: The quantity (for equity, commodity underlyings) / foreign amount (fx underlying)
  Allowable values: all positive real numbers
- Strike: The strike of the option
  Allowable values: all positive real numbers
- Underlyings: The basket of underlyings.
  Allowable values: for each underlying see 2.3.29
- OptionData: relevant are the long/short flag, the option type, the payoff type (must be set to MaxRainbow to identify the payoff), and the exercise date

(exactly one date must be given)
Allowable values: see 2.3.1 for the general structure of the option data node

- Settlement: the settlement date (optional, if not given defaults to the exercise date)
  Allowable values: each valid date.

The representation as a scripted trade is as follows:

```
<Trade id="MinRainbowOption#1">
  <TradeType>ScriptedTrade</TradeType>
  <Envelope/>
   <MinRainbowOptionData>
    <Expiry type="event">2020-02-15</Expiry>
    <Settlement type="event">2020-02-20</Settlement>
    <PutCall type="optionType">Call</PutCall>
    <LongShort type="longShort">Long</LongShort>
    <Notional type="number">1</Notional>
    <Strike type="number">2000</Strike>
    <Underlyings type="index">
      <Value>EQ-RIC:.SPX</Value>
      <Value>EQ-RIC:.STOXX50E</Value>
    </Underlyings>
    <Weights type="number">
      <Value>1.0</Value>
      <Value>1.0</Value>
    </Weights>
    <PayCcy type="currency">USD</PayCcy>
  </MinRainbowOptionData>
</Trade>
```

The MinRainbowOption script referenced in the trade above is shown in Listing 143.

The meanings and allowable values of the elements in the `MinRainbowOptionData` node are given below, with data type indicated in square brackets.

- [event] `Expiry`: Option expiry date.
  Allowable values: See `Date` in Table 13.
- [event] `Settlement`: Option settlement date.
  Allowable values: See `Date` in Table 13.
- [optionType] `PutCall`: Option type with
  Allowable values *Call, Put.*
- [longShort] `LongShort`: Position type, *Long* if we buy, *Short* if we sell.
  Allowable values: *Long, Short.*
- [number] `Notional`: Quantity multiplier applied to the basket price
  Allowable values: Any positive real number.
- [number] `Strike`: Strike price in PayCcy (see below)
  Allowable values: Any positive real number.
- [index] `Underlyings`: List of underlying indices enclosed by `<Value>` and `</Value>` tags.
  Allowable values: See ore/Docs/ScriptedTrade's Index Section for allowable values.
- [number] `Weights`: List of weights applied to each of the underlying prices, given in the same order as the Underlyings above, each weight enclosed by `<Value>`

and </Value> tags.
Allowable values: Any positive real number.

- [currency] PayCcy: The payment currency. For FX, where the underlying is provided in the form FX-SOURCE-CCY1-CCY2 (see Table 21) this should be CCY2. If CCY1 or the currency of the underlying (for EQ and COMM underlyings), this will result in a quanto payoff. Notice section "Payment Currency" in ore/Docs/ScriptedTrade.
  Allowable values: See Table 15 for allowable currency codes.

European Rainbow Call Spread Option

European rainbow call spread options are represented as *scripted trades*, refer to the scripted trade documentation in ore/Docs/ScriptedTrade for an introduction.

Trade input and the payoff script are described below.

```
<TradeType>ScriptedTrade</TradeType>
<Envelope/>
<EuropeanRainbowCallSpreadOptionData>
  <Expiry type="event">2020-02-15</Expiry>
  <Settlement type="event">2020-02-20</Settlement>
  <LongShort type="longShort">Long</LongShort>
  <Notional type="number">1000000</Notional>
  <Underlyings type="index">
    <Value>EQ-RIC:.SPX</Value>
    <Value>EQ-RIC:.STOXX50E</Value>
  </Underlyings>
  <InitialStrikes type="number">
    <Value>2100</Value>
    <Value>3000</Value>
  </InitialStrikes>
  <Weights type="number">
    <Value>0.8</Value>
    <Value>0.3</Value>
  </Weights>
  <Floor type="number">0.02</Floor>
  <Cap type="number">0.10</Cap>
  <PayCcy type="currency">USD</PayCcy>
</EuropeanRainbowCallSpreadOptionData>
```

The EuropeanRainbowCallSpreadOption script referenced in the trade above is shown in listing 144.

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

- [event] Expiry: Option expiry date.
  Allowable values: See Date in Table 13.
- [event] Settlement: Option settlement date.
  Allowable values: See Date in Table 13.
- [longShort] LongShort: long short flag.
  Allowable values: Long, Short
- [number] Notional: The notional.
  Allowable values: A real number.
- [index] Underlyings: The underlyings.

Allowable values: See ore/Docs/ScriptedTrade's Index Section for allowable values.

- [number] `InitialStrikes`: The initial strikes of the underlyings.
  Allowable values: A real number for each underlying.
- [number] `Weights`: The weights for the best, second best, ..., worst performing underlying.
  Allowable values: A real number for each rank.
- [number] `Floor`: The floor. If no floor applies, use e.g. -1E10. Allowable values: A real number.
- [number] `Cap`: The floor. If no floor applies, use e.g. 1E10. Allowable values: A real number.
- [currency] `PayCcy`: The payment currency. For FX, where the underlying is provided in the form `FX-SOURCE-CCY1-CCY2` (see Table 21) this should be `CCY2`. If `CCY1` or the currency of the underlying (for EQ and COMM underlyings), this will result in a quanto payoff. Notice section "Payment Currency" in ore/Docs/ScriptedTrade.
  Allowable values: See Table 15 for allowable currency codes.

Rainbow Call Spread Barrier Option

A rainbow call spread barrier option is an extension of the European Rainbow Call Spread Option, and is represented as *scripted trades*, refer to the scripted trade documentation in ore/Docs/ScriptedTrade for an introduction.

Trade input and the payoff script are described below.

```
<TradeType>ScriptedTrade</TradeType>
<Envelope>
  <CounterParty>CPTY_A</CounterParty>
  <NettingSetId>CPTY_A</NettingSetId>
  <AdditionalFields/>
</Envelope>
<RainbowCallSpreadBarrierOptionData>
  <Expiry type="event">2020-02-15</Expiry>
  <Settlement type="event">2020-02-20</Settlement>
  <LongShort type="longShort">Long</LongShort>
  <Notional type="number">1000000</Notional>
  <Underlyings type="index">
    <Value>EQ-RIC:.SPX</Value>
    <Value>EQ-RIC:.STOXX50E</Value>
  </Underlyings>
  <InitialPrices type="number">
    <Value>2100</Value>
    <Value>3000</Value>
  </InitialPrices>
  <Weights type="number">
    <Value>0.8</Value>
    <Value>0.3</Value>
  </Weights>
  <Strike type="number">1.0</Strike>
  <Floor type="number">0.02</Floor>
  <Cap type="number">0.10</Cap>
  <Gearing type="number">1.0</Gearing>
  <BermudanBarrier type="bool">false</BermudanBarrier>
  <BarrierLevel type="number">1000.0</BarrierLevel>
  <BarrierSchedule type="event">
    <ScheduleData>
      <Rules>
```

```
        <StartDate>2018-12-31</StartDate>
        <EndDate>2020-02-15</EndDate>
        <Tenor>1M</Tenor>
        <Calendar>USD</Calendar>
        <Convention>ModifiedFollowing</Convention>
        <TermConvention>ModifiedFollowing</TermConvention>
        <Rule>Forward</Rule>
      </Rules>
    </ScheduleData>
  </BarrierSchedule>
  <PayCcy type="currency">USD</PayCcy>
</RainbowCallSpreadBarrierOptionData>
```

The EuropeanRainbowCallSpreadOption script referenced in the trade above is shown in listing 145.

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

- [event] `Expiry`: Option expiry date.
  Allowable values: See `Date` in Table 13.
- [event] `Settlement`: Option settlement date.
  Allowable values: See `Date` in Table 13.
- [longShort] `LongShort`: long short flag.
  Allowable values: *Long*, *Short*
- [number] `Notional`: The notional.
  Allowable values: A real number.
- [index] `Underlyings`: The underlyings.
  Allowable values: See ore/Docs/ScriptedTrade's Index Section for allowable values.
- [number] `InitialPrices`: The initial prices of the underlyings.
  Allowable values: A real number for each underlying.
- [number] `Weights`: The weights for the best, second best, ..., worst performing underlying.
  Allowable values: A real number for each rank.
- [number] `Strike`: The option strike price.
  Allowable values: Any number, as a percentage expressed in decimal form.
- [number] `Floor`: The floor. If no floor applies, use e.g. -1E10.
  Allowable values: A real number.
- [number] `Cap`: The floor. If no floor applies, use e.g. 1E10.
  Allowable values: A real number.
- [number] `Gearing`: The gearing/payoff multiplier, applied after the cap and/or floor.
  Allowable values: A real number.
- [bool] `BermudanBarrier`: Whether the KI barrier observation is Bermudan (*True*) or European (*False*).
  Allowable values: *True* or *False*.

- [number] `BarrierLevel`: The agreed knock-in barrier price level.
  Allowable values: Any number.
- [event] `BarrierSchedule`: If `BermudanBarrier` is *True*, this is the barrier observation schedule. If *False*, this sub-node is still required but will not be used.
  Allowable values: See Section 2.3.4.
- [currency] `PayCcy`: The payment currency. For FX, where the underlying is provided in the form `FX-SOURCE-CCY1-CCY2` (see Table 21) this should be `CCY2`. If `CCY1` or the currency of the underlying (for EQ and COMM underlyings), this will result in a quanto payoff. Notice section "Payment Currency" in ore/Docs/ScriptedTrade.
  Allowable values: See Table 15 `Currency` for allowable currency codes.

Asian Rainbow Call Spread Option

Asian rainbow call spread options are represented as *scripted trades*, refer to the scripted trade documentation in ore/Docs/ScriptedTrade for an introduction.

Trade input and the payoff script are described below.

```
<TradeType>ScriptedTrade</TradeType>
<Envelope/>
<AsianRainbowCallSpreadOptionData>
  <Expiry type="event">2020-02-15</Expiry>
  <AveragingDates type="event">
    <ScheduleData>
      <Dates>
        <Dates>
          <Date>2019-01-29</Date>
                .....
          <Date>2020-02-15</Date>
        </Dates>
        <Calendar>USD</Calendar>
        <Convention>MF</Convention>
      </Dates>
    </ScheduleData>
  </AveragingDates>
  <Settlement type="event">2020-02-20</Settlement>
  <LongShort type="longShort">Long</LongShort>
  <Notional type="number">1000000</Notional>
  <Underlyings type="index">
    <Value>EQ-RIC:.SPX</Value>
    <Value>EQ-RIC:.STOXX50E</Value>
  </Underlyings>
  <InitialStrikes type="number">
    <Value>2100</Value>
    <Value>3000</Value>
  </InitialStrikes>
  <Weights type="number">
    <Value>0.8</Value>
    <Value>0.3</Value>
  </Weights>
  <Floor type="number">0.02</Floor>
  <Cap type="number">0.10</Cap>
  <PayCcy type="currency">USD</PayCcy>
</AsianRainbowCallSpreadOptionData>
```

The AsianRainbowCallSpreadOption script referenced in the trade above is shown in Listing 146.

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

- [event] `Expiry`: Option expiry date.
  Allowable values: See `Date` in Table 13.
- [event] `AveragingDates`: Observation dates for calculating the final (average) price of each underlying. Allowable values: See Section 2.3.4.
- [event] `Settlement`: Option settlement date.
  Allowable values: See `Date` in Table 13.
- [longShort] `LongShort`: long short flag.
  Allowable values: Long, Short
- [number] `Notional`: The notional.
  Allowable values: A real number.
- [index] `Underlyings`: The underlyings.
  Allowable values: See ore/Docs/ScriptedTrade's Index Section for allowable values.
- [number] `InitialStrikes`: The initial strikes of the underlyings.
  Allowable values: A real number for each underlying.
- [number] `Weights`: The weights for the best, second best, ..., worst performing underlying.
  Allowable values: A real number for each rank.
- [number] `Floor`: The floor. If no floor applies, use e.g. -1E10. Allowable values: A real number.
- [number] `Cap`: The floor. If no floor applies, use e.g. 1E10. Allowable values: A real number.
- [currency] `PayCcy`: The payment currency. For FX, where the underlying is provided in the form `FX-SOURCE-CCY1-CCY2` (see Table 21) this should be `CCY2`. If `CCY1` or the currency of the underlying (for EQ and COMM underlyings), this will result in a quanto payoff. Notice section "Payment Currency" in ore/Docs/ScriptedTrade.
  Allowable values: See Table 15 for allowable currency codes.

Worst Performance Rainbow Option 01

A worst performance rainbow option 01 is represented as a *scripted trade*, refer to the scripted trade documentation in ore/Docs/ScriptedTrade for an introduction.

Trade input and the payoff script are described below.

```
<TradeType>ScriptedTrade</TradeType>
<Envelope>
  <CounterParty>CPTY_A</CounterParty>
  <NettingSetId>CPTY_A</NettingSetId>
  <AdditionalFields/>
</Envelope>
<WorstPerformanceRainbowOption01Data>
  <LongShort type="longShort">Long</LongShort>
  <Underlyings type="index">
    <Value>EQ-RIC:.STOXX50E</Value>
    <Value>EQ-RIC:.SPX</Value>
  </Underlyings>
  <InitialPrices type="number">
```

```
      <Value>5455.60</Value>
      <Value>500</Value>
    </InitialPrices>
    <Premium type="number">291264</Premium>
    <PremiumDate type="event">2020-03-11</PremiumDate>
    <Quantity type="number">72816000</Quantity>
    <PayoffMultiplier type="number">0.4625</PayoffMultiplier>
    <ObservationDate type="event">2020-09-04</ObservationDate>
    <SettlementDate type="event">2020-09-11</SettlementDate>
    <PayCcy type="currency">EUR</PayCcy>
  </WorstPerformanceRainbowOption01Data>
```

The WorstPerformanceRainbowOption01 script referenced in the trade above is shown in listing 147.

The payout formula from the long perspective, determined on the `ObservationDate`, is

$$Payout = \texttt{Quantity} * (worstPerformance - 1)$$

where $worstPerformance$ is the performance, i.e. S_T/S_0, of the worst-performing asset as of the final determination date T.

The meanings and allowable values for the `WorstPerformanceRainbowOption01Data` node below.

- [longShort] `LongShort`: Own party position in the option.
  Allowable values: *Long, Short.*
- [index] `Underlyings`: The basket of underlyings.
  Allowable values: See ore/Docs/ScriptedTrade's Index Section for allowable values.
- [number] `InitialPrices`: The agreed initial price for each basket underlying.
  Allowable values: Any positive number.
- [number] `Premium`: Total option premium amount in terms of the *PayCcy*
  Allowable values: See Table 15 `Currency`.
- [event] `PremiumDate`: The premium payment date.
  Allowable values: See `Date` in Table 13.
- [number] `Quantity`: A quantity multiplier applied to the option payoff.
  Allowable values: Any number.
- [number] `PayoffMultiplier`: A factor that is multiplied to the option payoff when the option buyer incurs a negative net cash flow, i.e. when the performance of the worst-performing asset is less than 1.
  Allowable values: Any number, as a percentage expressed in decimal form.
- [event] `ObservationDate`: The date on which the final levels of the assets are determined.
  Allowable values: See `Date` in Table 13.
- [event] `SettlementDate`: The settlement date for the option payoff.
  Allowable values: See `Date` in Table 13.
- [currency] `PayCcy`: The payment currency. For FX, where the underlying is provided in the form `FX-SOURCE-CCY1-CCY2` (see Table 21) this should be `CCY2`.

If `CCY1` or the currency of the underlying (for EQ and COMM underlyings), this will result in a quanto payoff. Notice section "Payment Currency" in ore/Docs/ScriptedTrade.
Allowable values: See Table 15 for allowable currency codes.

Worst Performance Rainbow Option 02

A worst performance rainbow option 02 is represented as a *scripted trade*, refer to the scripted trade documentation in ore/Docs/ScriptedTrade for an introduction.

Trade input and the payoff script are described below.

```
<TradeType>ScriptedTrade</TradeType>
<Envelope>
  <CounterParty>CPTY_A</CounterParty>
  <NettingSetId>CPTY_A</NettingSetId>
  <AdditionalFields/>
</Envelope>
<WorstPerformanceRainbowOption02Data>
  <LongShort type="longShort">Long</LongShort>
  <Underlyings type="index">
    <Value>EQ-RIC:.STOXX50E</Value>
    <Value>EQ-RIC:.SPX</Value>
  </Underlyings>
  <InitialPrices type="number">
    <Value>4890.00</Value>
    <Value>108.84</Value>
  </InitialPrices>
  <Premium type="number">1731</Premium>
  <PremiumDate type="event">2020-02-27</PremiumDate>
  <Quantity type="number">90000000</Quantity>
  <PayoffMultiplier type="number">1.7</PayoffMultiplier>
  <Floor type="number">-0.05</Floor>
  <ObservationDate type="event">2020-11-13</ObservationDate>
  <SettlementDate type="event">2020-11-27</SettlementDate>
  <PayCcy type="currency">USD</PayCcy>
</WorstPerformanceRainbowOption02Data>
```

The WorstPerformanceRainbowOption02 script referenced in the trade above is shown in listing 148.

The payout formula from the long perspective, determined on the `ObservationDate`, is as follows, where $worstPerformance$ is the performance, i.e. S_T/S_0, of the worst-performing asset as of the final determination date T:

If $worstPerformance > 1$, receive

$$Payout = \texttt{Quantity} * \texttt{PayoffMultiplier} * (worstPerformance - 1).$$

If $worstPerformance \leq 1$, pay

$$Payout = \texttt{Quantity} * \max(\texttt{Floor}, worstPerformance - 1)$$

The meanings and allowable values for the `WorstPerformanceRainbowOption02Data` node below.

- [longShort] `LongShort`: Own party position in the option.
  Allowable values: *Long, Short.*

- [index] `Underlyings`: The basket of underlyings.
  Allowable values: See ore/Docs/ScriptedTrade's Index Section for allowable values.
- [number] `InitialPrices`: The agreed initial price for each basket underlying.
  Allowable values: Any positive number.
- [number] `Premium`: Total option premium amount in terms of the *PayCcy*
  Allowable values: See Table 15 `Currency`.
- [event] `PremiumDate`: The premium payment date.
  Allowable values: See `Date` in Table 13.
- [number] `Quantity`: A quantity multiplier applied to the option payoff.
  Allowable values: Any number.
- [number] `PayoffMultiplier`: A factor that is multiplied to the option payoff when the option buyer incurs a positive net cash flow, i.e. when the performance of the worst-performing asset is greater than 1.
  Allowable values: Any number, as a percentage expressed in decimal form.
- [number] `Floor`: The maximum loss that the option buyer can incur.
  Allowable values: Any non-positive number, as a percentage expressed in decimal form.
- [event] `ObservationDate`: The date on which the final levels of the assets are determined.
  Allowable values: See `Date` in Table 13.
- [event] `SettlementDate`: The settlement date for the option payoff.
  Allowable values: See `Date` in Table 13.
- [currency] `PayCcy`: The payment currency. For FX, where the underlying is provided in the form `FX-SOURCE-CCY1-CCY2` (see Table 21) this should be `CCY2`. If `CCY1` or the currency of the underlying (for EQ and COMM underlyings), this will result in a quanto payoff. Notice section "Payment Currency" in ore/Docs/ScriptedTrade.
  Allowable values: See Table 15 `Currency` for allowable currency codes.

Worst Performance Rainbow Option 03

A worst performance rainbow option 03 is an extension of the Worst Performance Rainbow Option 01, and is represented as a *scripted trade*, refer to the scripted trade documentation in ore/Docs/ScriptedTrade for an introduction.

Trade input and the payoff script are described below.

```
<TradeType>ScriptedTrade</TradeType>
<Envelope>
  <CounterParty>CPTY_A</CounterParty>
  <NettingSetId>CPTY_A</NettingSetId>
  <AdditionalFields/>
</Envelope>
<WorstPerformanceRainbowOption03Data>
  <LongShort type="longShort">Long</LongShort>
  <Underlyings type="index">
    <Value>EQ-RIC:.STOXX50E</Value>
```

```
      <Value>EQ-RIC:.SPX</Value>
    </Underlyings>
    <InitialPrices type="number">
      <Value>5455.60</Value>
      <Value>500</Value>
    </InitialPrices>
    <Premium type="number">291264</Premium>
    <PremiumDate type="event">2020-03-11</PremiumDate>
    <Strike type="number">1.0</Strike>
    <Quantity type="number">72816</Quantity>
    <PayoffMultiplier type="number">2.5</PayoffMultiplier>
    <Cap type="number">100.0</Cap>
    <Floor type="number">-100.0</Floor>
    <BermudanBarrier type="bool">false</BermudanBarrier>
    <BarrierLevel type="number">0.7</BarrierLevel>
    <BarrierSchedule type="event">
      <ScheduleData>
        <Rules>
          <StartDate>2020-03-11</StartDate>
          <EndDate>2020-09-04</EndDate>
          <Tenor>1D</Tenor>
          <Calendar>USD</Calendar>
          <Convention>ModifiedFollowing</Convention>
          <TermConvention>ModifiedFollowing</TermConvention>
          <Rule>Forward</Rule>
        </Rules>
      </ScheduleData>
    </BarrierSchedule>
    <ObservationDate type="event">2020-09-04</ObservationDate>
    <SettlementDate type="event">2020-09-11</SettlementDate>
    <PayCcy type="currency">EUR</PayCcy>
  </WorstPerformanceRainbowOption03Data>
```

The WorstPerformanceRainbowOption03 script referenced in the trade above is shown in listing 149.

The payout formula from the long perspective, determined on the `ObservationDate`, is as follows, where $worstPerformance$ is the performance, i.e. S_T/S_0, of the worst-performing asset as of the final determination date T:

If $worstPerformance \geq$ `Strike`, receive

$$Payout = \texttt{Quantity} * \min\big(\texttt{Cap}, \max(\texttt{Floor}, worstPerformance - \texttt{Strike})\big).$$

If $worstPerformance <$ `Strike`, pay

$$Payout = \texttt{Quantity}*\texttt{PayoffMultiplier}*\min\big(\texttt{Cap}, \max(\texttt{Floor}, worstPerformance - \texttt{Strike})\big).$$

The above payouts are conteningent on a knock-in event. If no knock-in has occurred, the payout is zero.

The meanings and allowable values for the `WorstPerformanceRainbowOption03Data` node below.

- [longShort] `LongShort`: Own party position in the option.
  Allowable values: *Long, Short*.
- [index] `Underlyings`: The basket of underlyings.
  Allowable values: See ore/Docs/ScriptedTrade's Index Section for allowable values.
- [number] `InitialPrices`: The agreed initial price for each basket underlying.
  Allowable values: Any positive number.

- [number] `Premium`: Total option premium amount in terms of the *PayCcy*
  Allowable values: See Table 15 `Currency`.
- [event] `PremiumDate`: The premium payment date.
  Allowable values: See `Date` in Table 13.
- [number] `Strike`: The option strike price.
  Allowable values: Any number, as a percentage expressed in decimal form.
- [number] `Quantity`: A quantity multiplier applied to the option payoff.
  Allowable values: Any number.
- [number] `PayoffMultiplier`: A factor that is multiplied to the option payoff when the option buyer incurs a positive net cash flow, i.e. when the performance of the worst-performing asset is greater than 1.
  Allowable values: Any number, as a percentage expressed in decimal form.
- [number] `Cap`: The maximum profit that the option buyer can receive.
  Allowable values: Any number, as a percentage expressed in decimal form.
- [number] `Floor`: The maximum loss that the option buyer can incur.
  Allowable values: Any number, as a percentage expressed in decimal form.
- [bool] `BermudanBarrier`: Whether the KI barrier observation is Bermudan (*True*) or European (*False*).
  Allowable values: *True* or *False*.
- [number] `BarrierLevel`: The agreed knock-in barrier price level. For example, in the case of a Bermudan barrier, if a knock-in is set to occur when one of the underlying prices falls below 70% of its initial price, then the appropriate value is *0.7*, as in the sample trade input above.
  Allowable values: Any number, as a percentage of the `InitialPrices` expressed in decimal form.
- [event] `BarrierSchedule`: If `BermudanBarrier` is *True*, this is the barrier observation schedule. If *False*, this sub-node is still required but will not be used.
  Allowable values: See Section 2.3.4.
- [event] `ObservationDate`: The date on which the final levels of the assets are determined.
  Allowable values: See `Date` in Table 13.
- [event] `SettlementDate`: The settlement date for the option payoff.
  Allowable values: See `Date` in Table 13.
- [currency] `PayCcy`: The payment currency. For FX, where the underlying is provided in the form `FX-SOURCE-CCY1-CCY2` (see Table 21) this should be `CCY2`. If `CCY1` or the currency of the underlying (for EQ and COMM underlyings), this will result in a quanto payoff. Notice section "Payment Currency" in ore/Docs/ScriptedTrade.
  Allowable values: See Table 15 for allowable currency codes.

Worst Performance Rainbow Option 04

A worst performance rainbow option 04 is represented as a *scripted trade*, refer to the scripted trade documentation in ore/Docs/ScriptedTrade for an introduction.

Trade input and the payoff script are described below.

```
<TradeType>ScriptedTrade</TradeType>
<Envelope>
  <CounterParty>CPTY_A</CounterParty>
  <NettingSetId>CPTY_A</NettingSetId>
  <AdditionalFields/>
</Envelope>
<WorstPerformanceRainbowOption04Data>
  <LongShort type="longShort">Long</LongShort>
  <Underlyings type="index">
    <Value>EQ-RIC:.STOXX50E</Value>
    <Value>EQ-RIC:.SPX</Value>
  </Underlyings>
  <InitialPrices type="number">
    <Value>5455.60</Value>
    <Value>500</Value>
  </InitialPrices>
  <Premium type="number">291264</Premium>
  <PremiumDate type="event">2020-03-11</PremiumDate>
  <Strike type="number">1.0</Strike>
  <Quantity type="number">72816</Quantity>
  <PayoffMultiplier type="number">2.5</PayoffMultiplier>
  <Cap type="number">100.0</Cap>
  <Floor type="number">-100.0</Floor>
  <BermudanBarrier type="bool">false</BermudanBarrier>
  <BarrierLevel type="number">0.7</BarrierLevel>
  <BarrierSchedule type="event">
    <ScheduleData>
      <Rules>
        <StartDate>2020-03-11</StartDate>
        <EndDate>2020-09-04</EndDate>
        <Tenor>1D</Tenor>
        <Calendar>USD</Calendar>
        <Convention>ModifiedFollowing</Convention>
        <TermConvention>ModifiedFollowing</TermConvention>
        <Rule>Forward</Rule>
      </Rules>
    </ScheduleData>
  </BarrierSchedule>
  <ObservationDate type="event">2020-09-04</ObservationDate>
  <SettlementDate type="event">2020-09-11</SettlementDate>
  <PayCcy type="currency">EUR</PayCcy>
</WorstPerformanceRainbowOption04Data>
```

The WorstPerformanceRainbowOption04 script referenced in the trade above is shown in listing 150.

The payout formula, determined on the `ObservationDate`, is as follows, where $worstPerformance$ is the performance, i.e. S_T/S_0, of the worst-performing asset as of the final determination date T:

If a knock-in event was triggered:

$$Payout = \texttt{Quantity} * (worstPerformance - \texttt{Strike}).$$

If no knock-in event was triggered:

$$Payout = \texttt{Quantity} * \min\Big(\texttt{Cap}, \max\big(\texttt{Floor}, \texttt{PayoffMultiplier} * (worstPerformance - \texttt{Strike})\big)\Big).$$

From the long perspective, the above amounts are received if they are positive, and paid out from the short perspective.

The meanings and allowable values for the `WorstPerformanceRainbowOption04Data` node below.

- [longShort] `LongShort`: Own party position in the option.
  Allowable values: *Long, Short*.
- [index] `Underlyings`: The basket of underlyings.
  Allowable values: See ore/Docs/ScriptedTrade's Index Section for allowable values.
- [number] `InitialPrices`: The agreed initial price for each basket underlying.
  Allowable values: Any positive number.
- [number] `Premium`: Total option premium amount in terms of the *PayCcy*
  Allowable values: See Table 15 `Currency`.
- [event] `PremiumDate`: The premium payment date.
  Allowable values: See `Date` in Table 13.
- [number] `Strike`: The option strike price.
  Allowable values: Any number, as a percentage expressed in decimal form.
- [number] `Quantity`: A quantity multiplier applied to the option payoff.
  Allowable values: Any number.
- [number] `PayoffMultiplier`: A factor that is multiplied to the option payoff when no knock-in event has ocurred. This multiplier is applied before the cap and/or floor. Allowable values: Any number, as a percentage expressed in decimal form.
- [number] `Cap`: The maximum profit that the option buyer can receive when a knock-in event has occurred.
  Allowable values: Any number, as a percentage expressed in decimal form.
- [number] `Floor`: The maximum loss that the option buyer can incur when a knock-in event has occurred.
  Allowable values: Any number, as a percentage expressed in decimal form.
- [bool] `BermudanBarrier`: Whether the KI barrier observation is Bermudan (*True*) or European (*False*).
  Allowable values: *True* or *False*.
- [number] `BarrierLevel`: The agreed knock-in barrier price level. For example, in the case of a Bermudan barrier, if a knock-in is set to occur when one of the underlying prices falls below 70% of its initial price, then the appropriate value is *0.7*, as in the sample trade input above.
  Allowable values: Any number, as a percentage of the `InitialPrices` expressed in decimal form.
- [event] `BarrierSchedule`: If `BermudanBarrier` is *True*, this is the barrier observation schedule. If *False*, this sub-node is still required but will not be used.
  Allowable values: See Section 2.3.4.

- [event] `ObservationDate`: The date on which the final levels of the assets are determined.
  Allowable values: See `Date` in Table 13.
- [event] `SettlementDate`: The settlement date for the option payoff.
  Allowable values: See `Date` in Table 13.
- [currency] `PayCcy`: The payment currency. For FX, where the underlying is provided in the form `FX-SOURCE-CCY1-CCY2` (see Table 21) this should be `CCY2`. If `CCY1` or the currency of the underlying (for EQ and COMM underlyings), this will result in a quanto payoff. Notice section "Payment Currency" in ore/Docs/ScriptedTrade.
  Allowable values: See Table 15 for allowable currency codes.

Worst Performance Rainbow Option 05

A worst performance rainbow option 05 is represented as a *scripted trade*, refer to the scripted trade documentation in ore/Docs/ScriptedTrade for an introduction.

Trade input and the payoff script are described below.

```
<TradeType>ScriptedTrade</TradeType>
<Envelope>
  <CounterParty>CPTY_A</CounterParty>
  <NettingSetId>CPTY_A</NettingSetId>
  <AdditionalFields/>
</Envelope>
<WorstPerformanceRainbowOption05Data>
  <LongShort type="longShort">Long</LongShort>
  <PutCall type="optionType">Put</PutCall>
  <Underlyings type="index">
    <Value>EQ-RIC:.STOXX50E</Value>
    <Value>EQ-RIC:.SPX</Value>
  </Underlyings>
  <InitialPrices type="number">
    <Value>5455.60</Value>
    <Value>500</Value>
  </InitialPrices>
  <Premium type="number">291264</Premium>
  <PremiumDate type="event">2020-03-11</PremiumDate>
  <Strike type="number">1.0</Strike>
  <Quantity type="number">72816</Quantity>
  <BarrierType type="barrierType">DownIn</BarrierType>
  <BarrierLevel type="number">0.6</BarrierLevel>
  <ObservationDate type="event">2020-09-04</ObservationDate>
  <SettlementDate type="event">2020-09-11</SettlementDate>
  <PayCcy type="currency">EUR</PayCcy>
</WorstPerformanceRainbowOption05Data>
```

The WorstPerformanceRainbowOption05 script referenced in the trade above is shown in listing 151.

The payout formula, determined on the `ObservationDate`, is as follows, where $worstPerformance$ is the performance, i.e. S_T/S_0, of the worst-performing asset as of the final determination date T. The payout for a long put option is as follows:

If a knock-in event was triggered,

$$Payout = \texttt{Quantity} * \max\Big(\big(\texttt{Strike} - worstPerformance\big), 0\Big).$$

Otherwise, the payoff is zero.

The meanings and allowable values for the `WorstPerformanceRainbowOption05Data` node below.

- [longShort] `LongShort`: Own party position in the option.
  Allowable values: *Long, Short*.
- [optionType] `PutCall`: Option type. For FX, this should be *Call* if we buy `CCY1` and sell `CCY2`, *Put* if we buy `CCY2` and sell `CCY1` (where the `Underlying` is in the form `FX-SOURCE-CCY1-CCY2`).
- [index] `Underlyings`: The basket of underlyings.
  Allowable values: See ore/Docs/ScriptedTrade's Index Section for allowable values.
- [number] `InitialPrices`: The agreed initial price for each basket underlying.
  Allowable values: Any positive number.
- [number] `Premium`: Total option premium amount in terms of the *PayCcy*
  Allowable values: See Table 15 `Currency`.
- [event] `PremiumDate`: The premium payment date.
  Allowable values: See `Date` in Table 13.
- [number] `Strike`: The option strike price.
  Allowable values: Any number, as a percentage expressed in decimal form.
- [number] `Quantity`: A quantity multiplier applied to the option payoff.
  Allowable values: Any number.
- [barrierType] `BarrierType`: The knock-in barrier type. For trades with no barrier, set `BarrierType` to *UpIn* and `BarrierLevel` to zero (or any negative number).
  Allowable values: *DownIn, UpIn*.
- [number] `BarrierLevel`: The agreed European knock-in barrier level.
  Allowable values: Any number, as a percentage of the `InitialPrices` expressed in decimal form.
- [event] `ObservationDate`: The date on which the final levels of the assets are determined.
  Allowable values: See `Date` in Table 13.
- [event] `SettlementDate`: The settlement date for the option payoff.
  Allowable values: See `Date` in Table 13.
- [currency] `PayCcy`: The payment currency. For FX, where the underlying is provided in the form `FX-SOURCE-CCY1-CCY2` (see Table 21) this should be `CCY2`. If `CCY1` or the currency of the underlying (for EQ and COMM underlyings), this will result in a quanto payoff. Notice section "Payment Currency" in ore/Docs/ScriptedTrade.
  Allowable values: See Table 15 for allowable currency codes.

2.2.83 Exotic Variance and Volatility Derivatives

These are vanilla variance/volatility swaps and options on an underlying with some additional payoff features, such as knock-in/knock-out barrier/s, conditions for the accrual of variance, basket underlyings, optionality on the payoff, etc.

Exotic variance/volatility swaps and options are represented as *scripted trades*, refer to the scripted trade documentation in ore/Docs/ScriptedTrade for an introduction. Each of the supported variations and their payoff script name are shown in Table 6. All swaps have an optional cap/floor feature.

Variance/Volatility Product Variation	Payoff Script Name
Variance/Volatility Option	`VarianceOption`
Variance/Volatility Swap with KI/KO Barrier	`KIKOVarianceSwap`
Corridor Volatility/Variance Swap	`CorridorVarianceSwap`
Corridor Variance Swap with KI/KO Barrier	`KIKOCorridorVarianceSwap`
Conditional Variance/Volatility Swap	`ConditionalVarianceSwap01` `ConditionalVarianceSwap02`
Pairwise Variance Swap	`PairwiseVarianceSwap`
Variance Dispersion Swap	`VarianceDispersionSwap`
Corridor Variance Dispersion Swap	`CorridorVarianceDispersionSwap`
Corridor Variance Dispersion Swap with KO Barrier	`KOCorridorVarianceDispersionSwap`
Gamma Swap	`GammaSwap`
Basket Variance Swap	`BasketVarianceSwap`

Table 6: Exotic variance/volatility product types and associated script names.

Trade input and the associated payoff script are described in the following for 12 supported variations.

Variance Option

The traditional trade representation is as follows, using an EQ underlying in this example:

```
<Trade id="EQ_VarianceOption">
  <TradeType>ScriptedTrade</TradeType>
  <Envelope>
    .....
  </Envelope>
  <VarianceOptionData>
    <LongShort type="longShort">Long</LongShort>
    <PutCall type="optionType">Call</PutCall>
    <PremiumAmount type="number">0</PremiumAmount>
    <PremiumDate type="event">2020-11-26</PremiumDate>
    <Notional type="number">138000</Notional>
    <VarianceReference type="number">0.19</VarianceReference>
    <Strike type="number">0.19</Strike>
    <Underlying type="index">EQ-RIC:.SPX</Underlying>
    <ValuationSchedule type="event">
      <ScheduleData>
        <Rules>
          <StartDate>2020-11-26</StartDate>
          <EndDate>2021-09-18</EndDate>
          <Tenor>1D</Tenor>
          <Convention>Following</Convention>
          <TermConvention>Following</TermConvention>
          <Calendar>USA</Calendar>
```

```
          <Rule>Forward</Rule>
        </Rules>
      </ScheduleData>
    </ValuationSchedule>
    <SquaredPayoff type="bool">true</SquaredPayoff>
    <SettlementDate type="event">2021-09-22</SettlementDate>
    <PayCcy type="currency">USD</PayCcy>
  </VarianceOptionData>
</Trade>
```

The VarianceOption script referenced in the trade above is shown in listing 152

The payout formulas for a variance put and call option are:

$$PutPayoff = 100^2 * \frac{\texttt{Notional}}{2 * 100 * \texttt{VarianceReference}} * \max[\texttt{Strike} - \texttt{RealisedVariance}, 0],$$
$$CallPayoff = 100^2 * \frac{\texttt{Notional}}{2 * 100 * \texttt{VarianceReference}} * \max[\texttt{Strike} - \texttt{RealisedVariance}, 0].$$

The meanings and allowable values for the `VarianceOptionData` node below.

- [longShort] `LongShort`: Own party position in the option. *Long* corresponds to paying out on the fixed/strike variance (volatility) and receiving on the floating/realised variance (volatility). In other words, a long position has positive value if the realised variance (volatility) exceeds the variance (volatility) strike.
  Allowable values: *Long, Short.*
- [optionType] `PutCall`: Option type. For FX, this should be *Call* if we buy `CCY1` and sell `CCY2`, *Put* if we buy `CCY2` and sell `CCY1` (where the `Underlying` is in the form `FX-SOURCE-CCY1-CCY2`).
  Allowable values: *Put, Call*
- [number] `PremiumAmount`: The total option premium amount. Allowable values: Any non-negativenumber.
- [event] `PremiumDate`: The option premium payment date. Allowable values: See `Date` in Table 13.
- [number] `Notional`: The vega notional amount. If the option was struck in terms of a variance notional N_{var}, the corresponding vega notional is given by $N_{vol} = N_{var} \cdot 2 \cdot 100 \cdot K_{vol}$ (where K_{vol} is in absolute terms).
  Allowable values: Any non-negative number.
- [number] `Strike`: The volatility strike K_{vol} quoted in absolute terms. If the option was struck in terms of variance, the square root of that variance should be used here.
  Allowable values: Any non-negative number, as a percentage expressed in decimal form.
- [number] `VarianceReference`: The parameter used to convert the vega notional amount into the corresponding variance amount. Similar to the `Strike`, this should be quoted in absolute terms, e.g. if the (volatility) strike price is 20% and the variance reference is 32.4%, then the `Strike` is *0.20* and the `VarianceReference` should be *0.324*.

Allowable values: Any non-negative number, as a percentage expressed in decimal form.

- [index] `Underlying`: Underlying index.
  Allowable values: See ore/Docs/ScriptedTrade's Index Section for allowable values.
- [event] `ValuationSchedule`: The schedule defining the (daily) observation period for the variance accrual.
  Allowable values: See Section 2.3.4.
- [bool] `SquaredPayoff`: Flag indicating whether the trade is a variance option (*True*) or a volatility option (*False*).
  Allowable values: Boolean node, allowing *Y*, *N*, *1*, *0*, *true*, *false*, etc. The full set of allowable values is given in Table 29.
- [event] `SettlementDate`: The date on which the option payoff is settled.
  Allowable values: See `Date` in Table 13.
- [currency] `PayCcy`: The payment currency. For FX, where the underlying is provided in the form `FX-SOURCE-CCY1-CCY2` (see Table 21) this should be `CCY2`. If `CCY1` or the currency of the underlying (for EQ and COMM underlyings), this will result in a quanto payoff. Notice section "Payment Currency" in ore/Docs/ScriptedTrade.
  Allowable values: See Table 15 for allowable currency codes.

Variance Swap with KI/KO Barrier

The traditional trade representation is as follows, using an FX underlying in this example:

```
<Trade id="FX_VarianceSwap_KIKO">
  <TradeType>ScriptedTrade</TradeType>
  <Envelope>
    .....
  </Envelope>
  <KIKOVarianceSwapData>
    <LongShort type="longShort">Long</LongShort>
    <Strike type="number">0.02</Strike>
    <Notional type="number">50000</Notional>
    <Underlying type="index">FX-ECB-EUR-USD</Underlying>
    <ValuationSchedule type="event">
      <ScheduleData>
        <Rules>
          <StartDate>2018-12-31</StartDate>
          <EndDate>2019-05-05</EndDate>
          <Tenor>1D</Tenor>
          <Convention>F</Convention>
          <TermConvention>F</TermConvention>
          <Calendar>US</Calendar>
          <Rule>Forward</Rule>
        </Rules>
      </ScheduleData>
    </ValuationSchedule>
    <SquaredPayoff type="bool">true</SquaredPayoff>
    <BarrierType type="barrierType">UpIn</BarrierType>
    <BarrierLevel type="number">0</BarrierLevel>
    <Cap type="number">2.5</Cap>
    <Floor type="number">0</Floor>
    <SettlementDate type="event">2019-05-06</SettlementDate>
    <PayCcy type="currency">USD</PayCcy>
```

```
  </KIKOVarianceSwapData>
</Trade>
```

The KIKOVarianceSwap script referenced in the trade above is shown in listing 153

The payout formula when no knock-out has occurred is:

$$Payout = 100^2 * varianceAmount * \Big[\min\big(\max(realisedVariance^2, \texttt{Floor}), \texttt{Cap} \big) - \texttt{Strike}^2 \Big].$$

If a knock-out has occurred, the above amount is scaled by the number of days between the first variance accrual date and the date of knock-out, as a fraction of the total number of variance accrual dates.

The meanings and allowable values for the `KIKOVarianceSwapData` node below.

- [longShort] `LongShort`: Own party position in the swap. *Long* corresponds to paying out on the fixed/strike variance (volatility) and receiving on the floating/realised variance (volatility). In other words, a long position has positive value if the realised variance (volatility) exceeds the variance (volatility) strike.
  Allowable values: *Long, Short.*
- [number] `Strike`: The volatility strike K_{vol} of the variance swap quoted in absolute terms. If the swap was struck in terms of variance, the square root of that variance should be used here.
  Allowable values: Any non-negative number, as a percentage expressed in decimal form.
- [number] `Notional`: The vega notional amount. If the swap was struck in terms of a variance notional N_{var}, the corresponding vega notional is given by $N_{vol} = N_{var} \cdot 2 \cdot 100 \cdot K_{vol}$ (where K_{vol} is in absolute terms).
  Allowable values: Any non-negative number.
- [index] `Underlying`: Underlying index.
  Allowable values: See ore/Docs/ScriptedTrade's Index Section for allowable values.
- [event] `ValuationSchedule`: The schedule defining the (daily) observation period for the variance accrual.
  Allowable values: `See` Section 2.3.4.
- [bool] SquaredPayoff: Flag indicating whether the trade is a variance swap (*True*) or a volatility swap (*False*).
  Allowable values: Boolean node, allowing *Y*, *N*, *1*, *0*, *true*, *false*, etc. The full set of allowable values is given in Table 29.
- [barrierType] `BarrierType`: Whether the barrier is a knock-in (*DownIn* or *UpIn*) or a knock-out (*DownOut* or *UpOut*) barrier. For trades with no barrier, which is equivalent to a capped/floored variance swap, set `BarrierType` to *UpIn* and `BarrierLevel` to zero, as in the sample trade representation above.
  Allowable values: *DownIn, UpIn, DownOut, UpOut.*
- [number] `BarrierLevel`: The agreed knock-in/knock-out barrier price level.
  Allowable values: Any non-negative real number.

- [number] `Cap`: The cap on the realised variance (or volatility), as a factor of the `Strike`. For example, if `Cap` is 2.5, then the cap level will be $2.5^2 \times$ `Strike`^{2} for variance swaps, and $2.5 \times$ `Strike` for volatility swaps. For trades with no cap, set `Cap` to zero.
  Allowable values: Any non-negative number.
- [number] `Floor`: The floor on the realised variance (or volatility), as a factor of the `Strike`. For example, if `Floor` is 0.1, then the floor level will be $0.1^2 \times$ `Strike` for variance swaps, and $0.1 \times$ `Strike` for volatility swaps. For trades with no floor, set `Floor` to zero.
  Allowable values: Any non-negative number.
- [event] `SettlementDate`: The date on which the swap payoff is settled.
  Allowable values: See `Date` in Table 13.
- [currency] `PayCcy`: The payment currency. For FX, where the underlying is provided in the form `FX-SOURCE-CCY1-CCY2` (see Table 21) this should be `CCY2`. If `CCY1` or the currency of the underlying (for EQ and COMM underlyings), this will result in a quanto payoff. Notice section "Payment Currency" in ore/Docs/ScriptedTrade.
  Allowable values: See Table 15 for allowable currency codes.

Corridor Variance Swap

The traditional trade representation is as follows, using an FX underlying in this example:

```
<Trade id="FX_VarianceSwap_Corridor">
  <TradeType>ScriptedTrade</TradeType>
  <Envelope>
    .....
  </Envelope>
  <CorridorVarianceSwapData>
    <LongShort type="longShort">Long</LongShort>
    <Strike type="number">0.02</Strike>
    <Notional type="number">50000</Notional>
    <Underlying type="index">FX-ECB-EUR-USD</Underlying>
    <ValuationSchedule type="event">
      <ScheduleData>
        <Rules>
          <StartDate>2018-12-31</StartDate>
          <EndDate>2019-05-05</EndDate>
          <Tenor>1D</Tenor>
          <Convention>F</Convention>
          <TermConvention>F</TermConvention>
          <Calendar>US</Calendar>
          <Rule>Forward</Rule>
        </Rules>
      </ScheduleData>
    </ValuationSchedule>
    <SquaredPayoff type="bool">true</SquaredPayoff>
    <UpperBarrierLevel type="number">1000000000</UpperBarrierLevel>
    <LowerBarrierLevel type="number">0</LowerBarrierLevel>
    <CountBothObservations type="bool">true</CountBothObservations>
    <AccrualAdjustment type="bool">false</AccrualAdjustment>
    <Cap type="number">2.5</Cap>
    <Floor type="number">0</Floor>
    <SettlementDate type="event">2019-05-06</SettlementDate>
    <PayCcy type="currency">USD</PayCcy>
  </CorridorVarianceSwapData>
</Trade>
```

The CorridorVarianceSwap script referenced in the trade above is shown in listing 154

The payout formula is:

$$Payout = 100^2 * varianceAmount * \Big[\min\big(\max(realisedVariance^2, \texttt{Floor}), \texttt{Cap} \big) - \texttt{Strike}^2 \Big],$$

where the $realisedVariance$ only consists of variance accrual contributions for when the underlying price/level was trading within the window defined by the corridor.

The meanings and allowable values for the `CorridorVarianceSwapData` node below.

- [longShort] `LongShort`: Own party position in the swap. *Long* corresponds to paying out on the fixed/strike variance (volatility) and receiving on the floating/realised variance (volatility). In other words, a long position has positive value if the realised variance (volatility) exceeds the variance (volatility) strike.
  Allowable values: *Long, Short*.
- [number] `Strike`: The volatility strike K_{vol} of the variance swap quoted in absolute terms. If the swap was struck in terms of variance, the square root of that variance should be used here.
  Allowable values: Any non-negative number, as a percentage expressed in decimal form.
- [number] `Notional`: The vega notional amount. If the swap was struck in terms of a variance notional N_{var}, the corresponding vega notional is given by $N_{vol} = N_{var} \cdot 2 \cdot 100 \cdot K_{vol}$ (where K_{vol} is in absolute terms).
  Allowable values: Any non-negative number.
- [index] `Underlying`: Underlying index.
  Allowable values: See ore/Docs/ScriptedTrade's Index Section for allowable values.
- [event] `ValuationSchedule`: The schedule defining the (daily) observation period for the variance accrual.
  Allowable values: See Section 2.3.4.
- [bool] `SquaredPayoff`: Flag indicating whether the trade is a variance swap (*True*) or a volatility swap (*False*).
  Allowable values: Boolean node, allowing *Y*, *N*, *1*, *0*, *true*, *false*, etc. The full set of allowable values is given in Table 29.
- [number] `UpperBarrierLevel`: The agreed upper barrier price level.
  Allowable values: Any non-negative real number.
- [number] `LowerBarrierLevel`: The agreed lower barrier price level.
  Allowable values: Any non-negative real number.
- [bool] `CountBothObservations`: Whether the variance/volatility is accrued based on both the current and previous underlying prices falling within the corridor (*True*) or only on the previous underlying price (*False*).
  Allowable values: Boolean node, allowing *Y*, *N*, *1*, *0*, *true*, *false*, etc. The full set of allowable values is given in Table 29.

- [bool] `AccrualAdjustment`: Whether the strike will be scaled relative to the number of days that the underlying traded within the corridor. See $\widetilde{K}_{var}$ (for variance swaps) and $\widetilde{K}_{vol}$ (for volatility swaps) in section Capped/Floored Corridor Variance Swap of the Product Description for Exotic Variance and Volatility Swaps. Allowable values: Boolean node, allowing *Y*, *N*, *1*, *0*, *true*, *false*, etc.

- [number] `Cap`: The cap on the realised variance (or volatility), as a factor of the `Strike` after the accrual adjustment is applied (only when `AccrualAdjustment` is *True*). For example, if `Cap` is 2.5, then the cap level will be $2.5^2 \times \widetilde{K}_{var}$ for variance swaps, and $2.5 \times \widetilde{K}_{vol}$ for volatility swaps. For trades with no cap, set `Cap` to zero.
  Allowable values: Any non-negative number.

- [number] `Floor`: The floor on the realised variance (or volatility), as a factor of the `Strike` after the accrual adjustment is applied (only when `AccrualAdjustment` is *True*). For example, if `Floor` is 0.1, then the floor level will be $0.1^2 \times \widetilde{K}_{var}$ for variance swaps, and $0.1 \times \widetilde{K}_{var}$ for volatility swaps. For trades with no floor, set `Floor` to zero.
  Allowable values: Any non-negative number. The full set of allowable values is given in Table 29.

- [event] `SettlementDate`: The date on which the swap payoff is settled.
  Allowable values: See `Date` in Table 13.

- [currency] `PayCcy`: The payment currency. For FX, where the underlying is provided in the form `FX-SOURCE-CCY1-CCY2` (see Table 21) this should be `CCY2`. If `CCY1` or the currency of the underlying (for EQ and COMM underlyings), this will result in a quanto payoff. Notice section "Payment Currency" in ore/Docs/ScriptedTrade.
  Allowable values: See Table 15 for allowable currency codes.

Corridor Variance Swap with KI/KO Barrier

The traditional trade representation is as follows, using an EQ underlying in this example:

```
<Trade id="EQ_VarianceSwap_KIKO_Corridor">
  <TradeType>ScriptedTrade</TradeType>
  <Envelope>
     .....
  </Envelope>
  <KIKOCorridorVarianceSwapData>
    <LongShort type="longShort">Long</LongShort>
    <Strike type="number">0.19</Strike>
    <Notional type="number">1380</Notional>
    <Underlying type="index">EQ-RIC:.SPX</Underlying>
    <ValuationSchedule type="event">
      <ScheduleData>
        <Rules>
          <StartDate>2020-11-26</StartDate>
          <EndDate>2021-09-18</EndDate>
          <Tenor>1D</Tenor>
          <Convention>Following</Convention>
          <TermConvention>Following</TermConvention>
          <Calendar>USA</Calendar>
          <Rule>Forward</Rule>
```

```
        </Rules>
      </ScheduleData>
    </ValuationSchedule>
    <CorridorUpperBarrierLevel type="number">3469.18</CorridorUpperBarrierLevel>
    <CorridorLowerBarrierLevel type="number">2207.66</CorridorLowerBarrierLevel>
    <KIKOBarrierType type="barrierType">UpOut</KIKOBarrierType>
    <KIKOBarrierLevel type="number">3469.18</KIKOBarrierLevel>
    <CountBothObservations type="bool">true</CountBothObservations>
    <AccrualAdjustment type="bool">true</AccrualAdjustment>
    <Cap type="number">0</Cap>
    <Floor type="number">0</Floor>
    <SettlementSchedule type="event">
      <DerivedSchedule>
        <BaseSchedule>ValuationSchedule</BaseSchedule>
        <Shift>2D</Shift>
        <Calendar>USA</Calendar>
        <Convention>Following</Convention>
      </DerivedSchedule>
    </SettlementSchedule>
    <PayCcy type="currency">USD</PayCcy>
  </KIKOCorridorVarianceSwapData>
</Trade>
```

The KIKOCorridorVarianceSwap script referenced in the trade above is shown in listing 153

The meanings and allowable values for the `KIKOCorridorVarianceSwapData` node below.

- [longShort] `LongShort`: Own party position in the swap. *Long* corresponds to paying out on the fixed/strike variance and receiving on the floating/realised variance. In other words, a long position has positive value if the realised variance exceeds the variance strike.
  Allowable values: *Long, Short*.
- [number] `Strike`: The volatility strike K_{vol} of the variance swap quoted in absolute terms. If the swap was struck in terms of variance, the square root of that variance should be used here.
  Allowable values: Any non-negative number, as a percentage expressed in decimal form.
- [number] `Notional`: The vega notional amount. If the swap was struck in terms of a variance notional N_{var}, the corresponding vega notional is given by $N_{vol} = N_{var} \cdot 2 \cdot 100 \cdot K_{vol}$ (where K_{vol} is in absolute terms).
  Allowable values: Any non-negative number.
- [index] `Underlying`: Underlying index. Currently only EQ, FX and COMM underlyings are supported.
  Allowable values: See ore/Docs/ScriptedTrade's Index Section for allowable values.
- [event] `ValuationSchedule`: The schedule defining the (daily) observation period for the variance accrual.
  Allowable values: See Section 2.3.4.
- [number] `CorridorUpperBarrierLevel`: The agreed upper barrier price level for the corridor.
  Allowable values: Any non-negative real number.
- [number] `CorridorLowerBarrierLevel`: The agreed lower barrier price level for

the corridor.
Allowable values: Any non-negative real number.

- [barrierType] `KIKOBarrierType`: Whether the barrier is a knock-in (*DownIn* or *UpIn*) or a knock-out (*DownOut* or *UpOut*) barrier. For trades with no barrier, which is equivalent to a capped/floored corridor variance swap, set `KIKOBarrierType` to *UpIn* and `KIKOBarrierLevel` to zero, as in the sample trade representation above.
Allowable values: *DownIn, UpIn, DownOut, UpOut.*

- [number] `KIKOBarrierLevel`: The agreed knock-in/knock-out barrier price level.
Allowable values: Any non-negative real number.

- [bool] `CountBothObservations`: Whether the variance is accrued based on both the current and previous underlying prices falling within the corridor (*True*) or only on the previous underlying price (*False*).
Allowable values: Boolean node, allowing *Y*, *N*, *1*, *0*, *true*, *false*, etc. The full set of allowable values is given in Table 29.

- [bool] `AccrualAdjustment`: Whether the strike will be scaled relative to the number of days that the underlying traded within the corridor. See $\widetilde{K}_{var}$ in section Capped/Floored Corridor Variance Swap of the Product Description for Exotic Variance and Volatility Swaps. Allowable values: Boolean node, allowing *Y*, *N*, *1*, *0*, *true*, *false*, etc.

- [number] `Cap`: The cap on the realised variance, as a factor of the `Strike`. For example, if `Cap` is 2.5, then the cap level will be $2.5^2 \times$ `Strike`^{2}. For trades with no cap, set `Cap` to zero.
Allowable values: Any non-negative number.

- [number] `Floor`: The floor on the realised variance, as a factor of the `Strike`. For example, if `Floor` is 0.1, then the floor level will be $0.1^2 \times$ `Strike`. For trades with no floor, set `Floor` to zero.
Allowable values: Any non-negative number.

- [event] `SettlementSchedule`: The settlement dates derived from the `ValuationSchedule` using a settlement lag to reflect settlement after a knock-out event, or after the final variance observation date if no knock-out occurs. If the barrier is knock-in, this represents the settlement date, which is the final valuation date plus the settlement lag.
Allowable values: See section 2.3.4 Schedule Data and Dates, or DerivedSchedule (see the scripted trade documentation in ore/Docs/ScriptedTrade).

- [currency] `PayCcy`: The payment currency. For FX, where the underlying is provided in the form `FX-SOURCE-CCY1-CCY2` (see Table 21) this should be `CCY2`. If `CCY1` or the currency of the underlying (for EQ and COMM underlyings), this will result in a quanto payoff. Notice section "Payment Currency" in ore/Docs/ScriptedTrade.
Allowable values: See Table 15 for allowable currency codes.

Conditional Variance Swap 01

The traditional trade representation is as follows, using an FX underlying in this example:

```
<Trade id="FX_VarianceSwap_Conditional_01">
  <TradeType>ScriptedTrade</TradeType>
  <Envelope>
    .....
  </Envelope>
  <ConditionalVarianceSwap01Data>
    <LongShort type="longShort">Long</LongShort>
    <Strike type="number">0.02</Strike>
    <Notional type="number">50000</Notional>
    <Underlying type="index">FX-ECB-EUR-USD</Underlying>
    <ValuationSchedule type="event">
      <ScheduleData>
        <Rules>
          <StartDate>2018-12-31</StartDate>
          <EndDate>2019-05-05</EndDate>
          <Tenor>1D</Tenor>
          <Convention>F</Convention>
          <TermConvention>F</TermConvention>
          <Calendar>US</Calendar>
          <Rule>Forward</Rule>
        </Rules>
      </ScheduleData>
    </ValuationSchedule>
    <SquaredPayoff type="bool">true</SquaredPayoff>
    <BarrierType type="barrierType">UpIn</BarrierType>
    <BarrierLevel type="number">0</BarrierLevel>
    <CountBothObservations type="bool">true</CountBothObservations>
    <Cap type="number">2.5</Cap>
    <Floor type="number">0</Floor>
    <AccrualAdjustment type="bool">false</AccrualAdjustment>
    <SettlementDate type="event">2019-05-06</SettlementDate>
    <PayCcy type="currency">USD</PayCcy>
  </ConditionalVarianceSwap01Data>
</Trade>
```

The ConditionalVarianceSwap01 script referenced in the trade above is shown in listing 156

The payout formula is:

$$Payout = 100^2 * varianceAmount * \Big[\min\big(\max(realisedVariance^2, \texttt{Floor}), \texttt{Cap} \big) - \texttt{Strike}^2 \Big],$$

where the $realisedVariance$ only consists of variance accrual contributions for when the underlying price/level was trading within the pre-defined window.

The meanings and allowable values for the `ConditionalVarianceSwapData01` node below.

- [longShort] `LongShort`: Own party position in the swap. *Long* corresponds to paying out on the fixed/strike variance (volatility) and receiving on the floating/realised variance (volatility). In other words, a long position has positive value if the realised variance (volatility) exceeds the variance (volatility) strike. Allowable values: *Long, Short.*
- [number] `Strike`: The volatility strike K_{vol} of the variance swap quoted in absolute terms. If the swap was struck in terms of variance, the square root of that variance should be used here.

Allowable values: Any non-negative number, as a percentage expressed in decimal form.

- [number] `Notional`: The vega notional amount. If the swap was struck in terms of a variance notional N_{var}, the corresponding vega notional is given by $N_{vol} = N_{var} \cdot 2 \cdot 100 \cdot K_{vol}$ (where K_{vol} is in absolute terms).
  Allowable values: Any non-negative number.
- [index] `Underlying`: Underlying index.
  Allowable values: See ore/Docs/ScriptedTrade's Index Section for allowable values.
- [event] `ValuationSchedule`: The schedule defining the (daily) observation period for the variance accrual.
  Allowable values: See Section 2.3.4.
- [bool] `SquaredPayoff`: Flag indicating whether the trade is a variance swap (*True*) or a volatility swap (*False*).
  Allowable values: Boolean node, allowing *Y*, *N*, *1*, *0*, *true*, *false*, etc. The full set of allowable values is given in Table 29.
- [barrierType] `BarrierType`: Whether the instrument is an up-variance swap (*UpIn* or *DownOut*) or a down-variance swap (*DownIn* or *UpOut*).
  Allowable values: *DownIn, UpIn, DownOut, UpOut.*
- [number] `BarrierLevel`: The agreed barrier price level.
  Allowable values: Any non-negative real number.
- [bool] `CountBothObservations`: Whether the variance/volatility is accrued based on both the current and previous underlying prices falling within the range (*True*) or only on the previous underlying price (*False*).
  Allowable values: Boolean node, allowing *Y*, *N*, *1*, *0*, *true*, *false*, etc. The full set of allowable values is given in Table 29.
- [bool] `AccrualAdjustment`: Whether the strike will be scaled relative to the number of days that the underlying traded within the range. See $\widetilde{K}_{var}$ (for variance swaps) and $\widetilde{K}_{vol}$ (for volatility swaps) in section Capped/Floored Conditional Variance Swap of the Product Description for Exotic Variance and Volatility Swaps. Allowable values: Boolean node, allowing *Y*, *N*, *1*, *0*, *true*, *false*, etc.
- [number] `Cap`: The cap on the realised variance (or volatility), as a factor of the `Strike` after the accrual adjustment is applied (only when `AccrualAdjustment` is *True*). For example, if `Cap` is 2.5, then the cap level will be $2.5^2 \times \widetilde{K}_{var}$ for variance swaps, and $2.5 \times \widetilde{K}_{vol}$ for volatility swaps. For trades with no cap, set `Cap` to zero.
  Allowable values: Any non-negative number.
- [number] `Floor`: The floor on the realised variance (or volatility), as a factor of the `Strike` after the accrual adjustment is applied (only when `AccrualAdjustment` is *True*). For example, if `Floor` is 0.1, then the floor level will be $0.1^2 \times \widetilde{K}_{var}$ for variance swaps, and $0.1 \times \widetilde{K}_{var}$ for volatility swaps. For trades with no floor, set `Floor` to zero.

Allowable values: Any non-negative number. The full set of allowable values is given in Table 29.

- [event] `SettlementDate`: The date on which the swap payoff is settled.
  Allowable values: See `Date` in Table 13.
- [currency] `PayCcy`: The payment currency. For FX, where the underlying is provided in the form `FX-SOURCE-CCY1-CCY2` (see Table 21) this should be `CCY2`. If `CCY1` or the currency of the underlying (for EQ and COMM underlyings), this will result in a quanto payoff. Notice section "Payment Currency" in ore/Docs/ScriptedTrade.
  Allowable values: See Table 15 for allowable currency codes.

Conditional Variance Swap 02

The traditional trade representation is as follows, using an FX underlying in this example:

```
<Trade id="FX_VarianceSwap_Conditional_02">
  <TradeType>ScriptedTrade</TradeType>
  <Envelope>
    .....
  </Envelope>
  <ConditionalVarianceSwap02Data>
    <LongShort type="longShort">Long</LongShort>
    <Strike type="number">0.02</Strike>
    <Notional type="number">50000</Notional>
    <VarianceReference type="number">0.03</VarianceReference>
    <Underlying type="index">FX-ECB-EUR-USD</Underlying>
    <ValuationSchedule type="event">
      <ScheduleData>
        <Rules>
          <StartDate>2018-12-31</StartDate>
          <EndDate>2019-05-05</EndDate>
          <Tenor>1D</Tenor>
          <Convention>F</Convention>
          <TermConvention>F</TermConvention>
          <Calendar>US</Calendar>
          <Rule>Forward</Rule>
        </Rules>
      </ScheduleData>
    </ValuationSchedule>
    <SquaredPayoff type="bool">true</SquaredPayoff>
    <BarrierType type="barrierType">UpIn</BarrierType>
    <BarrierLevel type="number">0</BarrierLevel>
    <CountBothObservations type="bool">true</CountBothObservations>
    <Cap type="number">2.5</Cap>
    <Floor type="number">0</Floor>
    <AccrualAdjustment type="bool">false</AccrualAdjustment>
    <SettlementDate type="event">2019-05-06</SettlementDate>
    <PayCcy type="currency">USD</PayCcy>
  </ConditionalVarianceSwap02Data>
</Trade>
```

The ConditionalVarianceSwap02 script referenced in the trade above is shown in listing 157

The payout formula is:

$$Payout = 100^2 * varianceAmount * \Big[\min\big(\max(realisedVariance^2, \texttt{Floor}), \texttt{Cap} \big) - \texttt{Strike}^2 \Big],$$

where the $realisedVariance$ only consists of variance accrual contributions for when the underlying price/level was trading within the pre-defined window, and the

"variance strike" used to obtain the varianceAmount is given by the variance reference strike (instead of the actual strike).

The meanings and allowable values for the `ConditionalVarianceSwapData02` node below.

- [longShort] `LongShort`: Own party position in the swap. *Long* corresponds to paying out on the fixed/strike variance (volatility) and receiving on the floating/realised variance (volatility). In other words, a long position has positive value if the realised variance (volatility) exceeds the variance (volatility) strike.
  Allowable values: *Long, Short.*
- [number] `Strike`: The volatility strike K_{vol} of the variance swap quoted in absolute terms. If the swap was struck in terms of variance, the square root of that variance should be used here.
  Allowable values: Any non-negative number, as a percentage expressed in decimal form.
- [number] `Notional`: The vega notional amount. If the swap was struck in terms of a variance notional N_{var}, the corresponding vega notional is given by $N_{vol} = N_{var} \cdot 2 \cdot 100 \cdot K_{vol}$ (where K_{vol} is in absolute terms).
  Allowable values: Any non-negative number.
- [number] `VarianceReference`: The (volatility) strike value used (in place of the actual strike) to scale down the vega notional in order to obtain the variance amount. If the swap was struck in terms of variance, the square root of that variance should be used here.
  Allowable values: Any non-negative number.
- [index] `Underlying`: Underlying index.
  Allowable values: See ore/Docs/ScriptedTrade's Index Section for allowable values.
- [event] `ValuationSchedule`: The schedule defining the (daily) observation period for the variance accrual.
  Allowable values: See Section 2.3.4.
- [bool] `SquaredPayoff`: Flag indicating whether the trade is a variance swap (*True*) or a volatility swap (*False*).
  Allowable values: Boolean node, allowing *Y*, *N*, *1*, *0*, *true*, *false*, etc. The full set of allowable values is given in Table 29.
- [barrierType] `BarrierType`: Whether the instrument is an up-variance swap (*UpIn* or *DownOut*) or a down-variance swap (*DownIn* or *UpOut*).
  Allowable values: *DownIn, UpIn, DownOut, UpOut.*
- [number] `BarrierLevel`: The agreed barrier price level.
  Allowable values: Any non-negative real number.
- [bool] `CountBothObservations`: Whether the variance/volatility is accrued based on both the current and previous underlying prices falling within the range (*True*) or only on the previous underlying price (*False*).
  Allowable values: Boolean node, allowing *Y*, *N*, *1*, *0*, *true*, *false*, etc. The full set of allowable values is given in Table 29.

- [bool] `AccrualAdjustment`: Whether the strike will be scaled relative to the number of days that the underlying traded within the range. See $\widetilde{K}_{var}$ (for variance swaps) and $\widetilde{K}_{vol}$ (for volatility swaps) in section Capped/Floored Conditional Variance Swap of the Product Description for Exotic Variance and Volatility Swaps. Allowable values: Boolean node, allowing *Y*, *N*, *1*, *0*, *true*, *false*, etc.

- [number] `Cap`: The cap on the realised variance (or volatility), as a factor of the `Strike` after the accrual adjustment is applied (only when `AccrualAdjustment` is *True*). For example, if `Cap` is 2.5, then the cap level will be $2.5^2 \times \widetilde{K}_{var}$ for variance swaps, and $2.5 \times \widetilde{K}_{vol}$ for volatility swaps. For trades with no cap, set `Cap` to zero.
  Allowable values: Any non-negative number.

- [number] `Floor`: The floor on the realised variance (or volatility), as a factor of the `Strike` after the accrual adjustment is applied (only when `AccrualAdjustment` is *True*). For example, if `Floor` is 0.1, then the floor level will be $0.1^2 \times \widetilde{K}_{var}$ for variance swaps, and $0.1 \times \widetilde{K}_{var}$ for volatility swaps. For trades with no floor, set `Floor` to zero.
  Allowable values: Any non-negative number. The full set of allowable values is given in Table 29.

- [event] `SettlementDate`: The date on which the swap payoff is settled.
  Allowable values: See `Date` in Table 13.

- [currency] `PayCcy`: The payment currency. For FX, where the underlying is provided in the form `FX-SOURCE-CCY1-CCY2` (see Table 21) this should be `CCY2`. If `CCY1` or the currency of the underlying (for EQ and COMM underlyings), this will result in a quanto payoff. Notice section "Payment Currency" in ore/Docs/ScriptedTrade.
  Allowable values: See Table 15 for allowable currency codes.

Pairwise Variance Swap

The `FxPairwiseVarianceSwap` and `EquityPairwiseVarianceSwap` trade types have trade data containers (respectively):

- `FxPairwiseVarianceSwapData`
- `EquityPairwiseVarianceSwapData`

Listing 158 shows the structure of example trades for an Equity underlying.

Listing 158: EquityPairwiseVarianceSwap data

```
<Trade id="EQ_VarianceSwap_Pairwise">
  <TradeType>EquityPairwiseVarianceSwap</TradeType>
  <Envelope>
    .......
  </Envelope>
  <EquityPairwiseVarianceSwapData>
    <LongShort>Long</LongShort>
    <Underlyings>
      <Value>EQ-RIC:.STOXX50E</Value>
      <Value>EQ-RIC:.SPX</Value>
    </Underlyings>
    <UnderlyingStrikes>
      <Value>0.33859119894055129</Value>
      <Value>0.39039467209479178</Value>
    </UnderlyingStrikes>
    <UnderlyingNotionals>
      <Value>577.75847822419905</Value>
      <Value>603.54235516510619</Value>
    </UnderlyingNotionals>
    <BasketNotional>1339.2898822637151</BasketNotional>
    <BasketStrike>0.31612972020991642</BasketStrike>
    <ValuationSchedule>
      <Rules>
      .......
      </Rules>
    </ValuationSchedule>
    <LaggedValuationSchedule>
      <Derived>
       .......
      </Derived>
    </LaggedValuationSchedule>
    <AccrualLag>3</AccrualLag>
    <PayoffLimit>5</PayoffLimit>
    <Cap>2.5</Cap>
    <Floor>0</Floor>
    <SettlementDate>2022-06-29</SettlementDate>
    <PayCcy>USD</PayCcy>
  </EquityPairwiseVarianceSwapData>
</Trade>
```

The payout formula is:

$$Payout = \min\Big(\max(equityAmount1 + equityAmount2 + equityAmountBasket, LowerLimit), UpperL$$

where

$$\begin{aligned} UpperLimit &= \texttt{PayoffLimit} * \big(|notional1| + |notional2|\big) \\ LowerLimit &= - \texttt{PayoffLimit} * \big(|notional1| + |notional2|\big) \end{aligned}$$

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

- `LongShort`: Own party position.
  Allowable values: *Long*, *Short*
- `Underlyings`: The basket of underlyings.

Allowable values: The format follows that of a scripted trade underlying. See the scripted trade documentation in ore/Docs/ScriptedTrade (section Data Node / Index) for allowable values.

- `UnderlyingStrikes`: The volatility strikes $K_{vol}^{1,2}$ of the underlyings quoted in absolute terms. If the swap was struck in terms of variance, the square roots of the variances should be used here.
  Allowable values: Any non-negative number, as a percentage expressed in decimal form.

- `UnderlyingNotionals`: The vega notional amount. If the swap was struck in terms of variance notionals N_{var}, the corresponding vega notionals are given by $N_{vol}^{1,2} = N_{var}^{1,2} \cdot 2 \cdot 100 \cdot K_{vol}^{1,2}$ (where $K_{vol}^{1,2}$ is in absolute terms).
  Allowable values: Any non-negative number.

- `BasketNotional`: The basket vega notional amount.
  Allowable values: Any non-negative number.

- `ValuationSchedule`: The start dates of the variance accrual schedule. This node can also be used to derive the variance accrual schedule (or vice versa) using a DerivedSchedule (e.g. with a two-day lag, *-2D*). For the standard accrual period lengths of one day, set `Shift` to *-1D*.
  Allowable values: See Section 2.3.4. If this uses a DerivedSchedule, the `Shift` must be less than or equal to *-1D*.

- `LaggedValuationSchedule`: The end dates of the variance accrual schedule. This node can be derived from the `ValuationSchedule` (or vice versa) using a DerivedSchedule (e.g. with a two-day lag, *2D*). For the standard accrual period lengths of one day, set `Shift` to *1D*.
  Allowable values: See section 2.3.4. If this uses a DerivedSchedule, the `Shift` must be greater than or equal to *1D*. If left blank or omitted, defaults to a derived schedule, with `Shift` equal to *1D*.

- `AccrualLag`: The length (in days) of each variance accrual period. For classic variance swaps, set this to *1*.
  Allowable values: Any integer greater than or equal to *1*. If left blank or omitted, defaults to *1*.

- `PayoffLimit`: The factor used to determine the maximum/minimum payoff under the swap. This corresponds to C in the final equation in the Product Description for Pairwise Variance Swap.
  Allowable values: Any non-negative number. If left blank or omitted, defaults to zero, i.e. no payoff limit.

- `Cap`: The cap on the realised variances, as a factor of the corresponding strikes. For example, if `Cap` is 2.5, then the cap level will be $2.5^2 \times \hat{K}_{var}$ for the basket realised variance. For trades with no cap, set `Cap` to zero.
  Allowable values: Any non-negative number.

- `Floor`: The floor on the realised variances, as a factor of the corresponding strikes. For example, if `Floor` is 0.1, then the floor level will be $0.1^2 \times \hat{K}_{var}$ for the basket realised variance. For trades with no floor, set `Floor` to zero.
  Allowable values: Any non-negative number.

- `SettlementDate`: The date on which the swap payoff is settled.
  Allowable values: See `Date` in Table 13.
- `PayCcy`: The settlement currency.
  Allowable values: See Table 15 `Currency`.

Pairwise variance swaps can alternatively be represented as *scripted trades*, refer to the scripted trade documentation in ore/Docs/ScriptedTrade for an introduction.

```
<Trade id="EQ_VarianceSwap_Pairwise">
  <TradeType>ScriptedTrade</TradeType>
  <Envelope>
    .....
  </Envelope>
  <PairwiseVarianceSwapData>
    <LongShort type="longShort">Long</LongShort>
    <Underlyings type="index">
      <Value>EQ-RIC:.STOXX50E</Value>
      <Value>EQ-RIC:.SPX</Value>
    </Underlyings>
    <UnderlyingStrikes type="number">
      <Value>0.3385911989405513</Value>
      <Value>0.3903946720947918</Value>
    </UnderlyingStrikes>
    <UnderlyingNotionals type="number">
      <Value>577.7584782241991</Value>
      <Value>603.5423551651062</Value>
    </UnderlyingNotionals>
    <BasketNotional type="number">1339.289882263715</BasketNotional>
    <BasketStrike type="number">0.3161297202099164</BasketStrike>
    <ValuationSchedule type="event">
      <ScheduleData>
        <Rules>
          <StartDate>2021-06-25</StartDate>
          <EndDate>2022-06-25</EndDate>
          <Tenor>3D</Tenor>
          <Convention>F</Convention>
          <TermConvention>F</TermConvention>
          <Calendar>US</Calendar>
          <Rule>Forward</Rule>
        </Rules>
      </ScheduleData>
    </ValuationSchedule>
    <LaggedValuationSchedule type="event">
      <DerivedSchedule>
        <BaseSchedule>ValuationSchedule</BaseSchedule>
        <Shift>2D</Shift>
        <Calendar>US</Calendar>
        <Convention>F</Convention>
      </DerivedSchedule>
    </LaggedValuationSchedule>
    <AccrualLag type="number">3</AccrualLag>
    <PayoffLimit type="number">5</PayoffLimit>
    <Cap type="number">2.5</Cap>
    <Floor type="number">0</Floor>
    <SettlementDate type="event">2022-06-29</SettlementDate>
    <PayCcy type="currency">USD</PayCcy>
  </PairwiseVarianceSwapData>
</Trade>
```

The PairwiseVarianceSwap script referenced in the trade above is shown in listing 159

The meanings and allowable values for the `PairwiseVarianceSwapData` node below.

- [longShort] `LongShort`: Own party position in the swap. *Long* corresponds to paying out on the fixed/strike variance (volatility) and receiving on the floating/realised variance (volatility). In other words, a long position has positive

value if the realised variance (volatility) exceeds the variance (volatility) strike.
Allowable values: *Long, Short*.

- [number] `UnderlyingStrikes`: The volatility strikes $K_{vol}^{1,2}$ of the underlyings quoted in absolute terms. If the swap was struck in terms of variance, the square roots of the variances should be used here.
  Allowable values: Any non-negative number, as a percentage expressed in decimal form.

- [number] `UnderlyingNotionals`: The vega notional amount. If the swap was struck in terms of variance notionals N_{var}, the corresponding vega notionals are given by $N_{vol}^{1,2} = N_{var}^{1,2} \cdot 2 \cdot 100 \cdot K_{vol}^{1,2}$ (where $K_{vol}^{1,2}$ is in absolute terms).
  Allowable values: Any non-negative number.

- [index] `Underlyings`: Underlying index.
  Allowable values: See ore/Docs/ScriptedTrade's Index Section for allowable values.

- [number] `BasketStrike`: The basket volatility strike $\hat{K}_{vol}$ quoted in absolute terms.
  Allowable values: Any non-negative number, as a percentage expressed in decimal form.

- [number] `BasketNotional`: The basket vega notional amount.
  Allowable values: Any non-negative number.

- [event] `ValuationSchedule`: The base schedule which defines the start dates of the variance accrual schedule. This node is also used to derive the variance accrual schedule.
  Allowable values: See Section 2.3.4.

- [event] `LaggedValuationSchedule`: The end dates of the variance accrual schedule. This can be derived from the `Schedule` using a DerivedSchedule (e.g. with a two-day lag, *2D*). For the standard accrual period lengths of one day, set `Shift` to *1D*.
  Allowable values: See section 2.3.4 Schedule Data and Dates, or DerivedSchedule (see the scripted trade documentation in ore/Docs/ScriptedTrade).

- [number] `AccrualLag`: The length (in days) of each variance accrual period. For classic variance swaps with no lag, set this to 1.
  Allowable values: Any integer greater than or equal to 1.

- [number] `PayoffLimit`: The factor used to determine the maximum/minimum payoff under the swap. This corresponds to C in the final equation in the Product Description for Pairwise Variance Swap.
  Allowable values: Any non-negative number.

- [number] `Cap`: The cap on the realised variances, as a factor of the corresponding strikes. For example, if `Cap` is 2.5, then the cap level will be $2.5^2 \times \hat{K}_{var}$ for the basket realised variance. For trades with no cap, set `Cap` to zero.
  Allowable values: Any non-negative number.

- [number] `Floor`: The floor on the realised variances, as a factor of the corresponding strikes. For example, if `Floor` is 0.1, then the floor level will be

$0.1^2 \times \hat{K}_{var}$ for the basket realised variance. For trades with no floor, set `Floor` to zero.
Allowable values: Any non-negative number.

- [event] `SettlementDate`: The date on which the swap payoff is settled.
  Allowable values: See `Date` in Table 13.

- [currency] `PayCcy`: The payment currency. For FX, where the underlying is provided in the form `FX-SOURCE-CCY1-CCY2` (see Table 21) this should be `CCY2`. If `CCY1` or the currency of the underlying (for EQ and COMM underlyings), this will result in a quanto payoff. Notice section "Payment Currency" in ore/Docs/ScriptedTrade.
  Allowable values: See Table 15 for allowable currency codes.

Variance Dispersion Swap

The traditional trade representation is as follows, using EQ underlyings in this example:

```
<Trade id="EQ_VarianceDispersionSwap">
  <TradeType>ScriptedTrade</TradeType>
  <Envelope>
    .....
  </Envelope>
  <VarianceDispersionSwapData>
    <LongShort type="longShort">Long</LongShort>
    <Underlyings1 type="index">
      <Value>EQ-RIC:HSBA.L</Value>
      <Value>EQ-RIC:MSFT.OQ</Value>
      <Value>EQ-RIC:AMZN.O</Value>
    </Underlyings1>
    <Weights1 type="number">
        .....
    </Weights1>
    <Strikes1 type="number">
        .....
    </Strikes1>
    <Spreads1 type="number">
        .....
    </Spreads1>
    <Notionals1 type="number">
        .....
    </Notionals1>
    <Caps1 type="number">
        .....
    </Caps1>
    <Floors1 type="number">
        .....
    </Floors1>
    <Underlyings2 type="index">
      <Value>EQ-RIC:.STOXX50E</Value>
      <Value>EQ-RIC:.SPX</Value>
    </Underlyings2>
    <Weights2 type="number">
        .....
    </Weights2>
    <Strikes2 type="number">
        .....
    </Strikes2>
    <Spreads2 type="number">
        .....
    </Spreads2>
    <Notionals2 type="number">
        .....
    </Notionals2>
    <Caps2 type="number">
```

```
      .....
    </Caps2>
    <Floors2 type="number">
      .....
    </Floors2>
    <ValuationSchedule type="event">
      <ScheduleData>
        .....
      </ScheduleData>
    </ValuationSchedule>
    <DividendAdjustment type="bool">false</DividendAdjustment>
    <SettlementDate type="event">2021-01-10</SettlementDate>
    <PayCcy type="currency">USD</PayCcy>
  </VarianceDispersionSwapData>
</Trade>
```

The VarianceDispersionSwap script referenced in the trade above is shown in listing 160

The meanings and allowable values for the `VarianceDispersionSwapData` node below.

- [longShort] `LongShort`: Own party position in the swap. *Long* corresponds to selling volatility on the `Underlyings1` basket and buying volatility on `Underlyings2`.
  Allowable values: *Long, Short.*
- [index] `Underlyings1`: The basket of underlyings whose volatility is bought in the *Long* position.
  Allowable values: For each underlying, see 2.3.29.
- [number] `Weights1`: List of weights applied to the final realised volatility of each underlying in the `Underlyings1` basket.
  Allowable values: Any positive number.
- [number] `Strikes1`: The volatility strike $K_{1,u,vol}$ of the variance swap for each underlying u in the `Underlyings1` basket, quoted in absolute terms. If the swap was struck in terms of variance, the square root of that variance should be used here.
  Allowable values: Any positive number, as a percentage expressed in decimal form.
- [number] `Spreads1`: Additional spread to the realised variance, for each underlying in the `Underlyings1` basket.
  Allowable values: Any real number.
- [number] `Notionals1`: For each underlying u in the `Underlyings1` basket, the vega notional amount. If the swap was struck in terms of variance notionals $N_{1,u,var}$, the corresponding vega notionals are given by $N_{1,u,vol} = N_{1,u,var} \cdot 2 \cdot 100 \cdot K_{1,u,vol}$ (where $K_{1,u,vol}$ is in absolute terms).
  Allowable values: Any real number.
- [number] `Caps1`: For each underlying u in the `Underlyings1` basket, the cap on the realised variance as a factor of the strike. For example, if the value in `Caps1` is 2.5, then the cap level will be $2.5^2 \times K_{1,u,\text{var}}$. For underlyings with no cap, set the value in `Caps1` to zero.
  Allowable values: Any non-negative number.
- [number] `Floors1`: For each underlying u in the `Underlyings1` basket, the floor

on the realised variance as a factor of the strike. For example, if the value in `Floors1` is 0.1, then the floor level will be $0.1^2 \times K_{1,u,\text{var}}$. For underlyings with no floor, set the value in `Floors1` to zero.
Allowable values: Any non-negative number.

- [index] `Underlyings2`: The basket of underlyings whose volatility is sold in the *Long* position.
  Allowable values: For each underlying, see 2.3.29.
- [number] `Weights2`: List of weights applied to the final realised volatility of each underlying in the `Underlyings2` basket.
  Allowable values: Any positive number.
- [number] `Strikes2`: The volatility strike $K_{2,u,vol}$ of the variance swap for each underlying u in the `Underlyings2` basket, quoted in absolute terms. If the swap was struck in terms of variance, the square root of that variance should be used here.
  Allowable values: Any positive number, as a percentage expressed in decimal form.
- [number] `Spreads2`: Additional spread to the realised variance, for each underlying in the `Underlyings2` basket.
  Allowable values: Any real number.
- [number] `Notionals2`: For each underlying u in the `Underlyings2` basket, the vega notional amount. If the swap was struck in terms of variance notionals $N_{2,u,var}$, the corresponding vega notionals are given by $N_{2,u,vol} = N_{2,u,var} \cdot 2 \cdot 100 \cdot K_{2,u,vol}$ (where $K_{2,u,vol}$ is in absolute terms).
  Allowable values: Any real number.
- [number] `Caps2`: For each underlying u in the `Underlyings2` basket, the cap on the realised variance as a factor of the strike. For example, if the value in `Caps2` is 2.5, then the cap level will be $2.5^2 \times K_{2,u,\text{var}}$. For underlyings with no cap, set the value in `Caps2` to zero.
  Allowable values: Any non-negative number.
- [number] `Floors2`: For each underlying u in the `Underlyings2` basket, the floor on the realised variance as a factor of the strike. For example, if the value in `Floors2` is 0.1, then the floor level will be $0.1^2 \times K_{2,u,\text{var}}$. For underlyings with no floor, set the value in `Floors2` to zero.
  Allowable values: Any non-negative number.
- [bool] `DividendAdjustment`: Whether or not the underlying price is adjusted for dividend payments. This feature is not yet supported. Allowable values: Boolean node, allowing *Y*, *N*, *1*, *0*, *true*, *false*, etc. The full set of allowable values is given in Table 29.
- [event] `ValuationSchedule`: The schedule defining the (daily) observation period for the variance accrual.
  Allowable values: See Section 2.3.4.
- [event] `SettlementDate`: The date on which the swap payoff is settled.
  Allowable values: See `Date` in Table 13.

- [currency] `PayCcy`: The payment currency. For FX, where the underlying is provided in the form `FX-SOURCE-CCY1-CCY2` (see Table 21) this should be `CCY2`. If `CCY1` or the currency of the underlying (for EQ and COMM underlyings), this will result in a quanto payoff. Notice section "Payment Currency" in ore/Docs/ScriptedTrade.
  Allowable values: See Table 15 for allowable currency codes.

Corridor Variance Dispersion Swap

This instrument is a variance dispersion swap with a corridor feature, where variance is accrued only when the the price of the first underlying of each pair of underlyings (between the first and second basket) falls within the 'corridor'. This means that the number of underlyings in each basket should be the same, and each pair is determined based on the order that they are provided in the trade XML, as in the example below.

The traditional trade representation is as follows, using EQ underlyings in this example:

```
<Trade id="EQ_VarianceDispersionSwap_Corridor">
  <TradeType>ScriptedTrade</TradeType>
  <Envelope>
    .....
  </Envelope>
  <CorridorVarianceDispersionSwapData>
    <LongShort type="longShort">Long</LongShort>
    <Weights type="number">
        .....
    </Weights>
    <Underlyings1 type="index">
      <Value>EQ-RIC:HSBA.L</Value>
      <Value>EQ-RIC:MSFT.OQ</Value>
      <Value>EQ-RIC:AMZN.O</Value>
    </Underlyings1>
    <Strikes1 type="number">
        .....
    </Strikes1>
    <Spreads1 type="number">
        .....
    </Spreads1>
    <Notionals1 type="number">
        .....
    </Notionals1>
    <Caps1 type="number">
        .....
    </Caps1>
    <Floors1 type="number">
        .....
    </Floors1>
    <Underlyings2 type="index">
      <Value>EQ-RIC:.STOXX50E</Value>
      <Value>EQ-RIC:.SPX</Value>
      <Value>EQ-RIC:.SPX</Value>
    </Underlyings2>
    <Strikes2 type="number">
        .....
    </Strikes2>
    <Spreads2 type="number">
        .....
    </Spreads2>
    <Notionals2 type="number">
        .....
    </Notionals2>
    <Caps2 type="number">
        .....
    </Caps2>
    <Floors2 type="number">
```

```
      .....
    </Floors2>
    <UpperBarrierLevels type="number">
      .....
    </UpperBarrierLevels>
    <LowerBarrierLevels type="number">
      .....
    </LowerBarrierLevels>
    <CountBothObservations type="bool">true</CountBothObservations>
    <AccrualAdjustment type="bool">true</AccrualAdjustment>
    <DividendAdjustment type="bool">false</DividendAdjustment>
    <ValuationSchedule type="event">
      <ScheduleData>
        .....
      </ScheduleData>
    </ValuationSchedule>
    <SettlementDate type="event">2021-01-10</SettlementDate>
    <PayCcy type="currency">USD</PayCcy>
  </CorridorVarianceDispersionSwapData>
</Trade>
```

The CorridorVarianceDispersionSwap script referenced in the trade above is shown in listing 161

The meanings and allowable values for the `CorridorVarianceDispersionSwapData` node below.

- [longShort] `LongShort`: Own party position in the swap. *Long* corresponds to selling volatility on the `Underlyings1` basket and buying volatility on `Underlyings2`.
  Allowable values: *Long, Short*.
- [number] `Weights`: List of weights applied to the final realised volatility of each underlying in the `Underlyings1` and `Underlyings2` baskets.
  Allowable values: Any positive number.
- [index] `Underlyings1`: The basket of underlyings whose volatility is bought in the *Long* position.
  Allowable values: For each underlying, see 2.3.29.
- [number] `Strikes1`: The volatility strike $K_{1,u,vol}$ of the variance swap for each underlying u in the `Underlyings1` basket, quoted in absolute terms. If the swap was struck in terms of variance, the square root of that variance should be used here.
  Allowable values: Any positive number, as a percentage expressed in decimal form.
- [number] `Spreads1`: Additional spread to the realised variance, for each underlying in the `Underlyings1` basket.
  Allowable values: Any real number.
- [number] `Notionals1`: For each underlying u in the `Underlyings1` basket, the vega notional amount. If the swap was struck in terms of variance notionals $N_{1,u,var}$, the corresponding vega notionals are given by $N_{1,u,vol} = N_{1,u,var} \cdot 2 \cdot 100 \cdot K_{1,u,vol}$ (where $K_{1,u,vol}$ is in absolute terms).
  Allowable values: Any real number.
- [number] `Caps1`: For each underlying u in the `Underlyings1` basket, the cap on the realised variance as a factor of the strike. For example, if the value in `Caps1`

is 2.5, then the cap level will be $2.5^2 \times K_{1,u,\text{var}}$. For underlyings with no cap, set the value in `Caps1` to zero.
Allowable values: Any non-negative number.

- [number] `Floors1`: For each underlying u in the `Underlyings1` basket, the floor on the realised variance as a factor of the strike. For example, if the value in `Floors1` is 0.1, then the floor level will be $0.1^2 \times K_{1,u,\text{var}}$. For underlyings with no floor, set the value in `Floors1` to zero.
Allowable values: Any non-negative number.

- [index] `Underlyings2`: The basket of underlyings whose volatility is sold in the *Long* position.
Allowable values: For each underlying, see 2.3.29.

- [number] `Strikes2`: The volatility strike $K_{2,u,vol}$ of the variance swap for each underlying u in the `Underlyings2` basket, quoted in absolute terms. If the swap was struck in terms of variance, the square root of that variance should be used here.
Allowable values: Any positive number, as a percentage expressed in decimal form.

- [number] `Spreads2`: Additional spread to the realised variance, for each underlying in the `Underlyings2` basket.
Allowable values: Any real number.

- [number] `Notionals2`: For each underlying u in the `Underlyings2` basket, the vega notional amount. If the swap was struck in terms of variance notionals $N_{2,u,var}$, the corresponding vega notionals are given by $N_{2,u,vol} = N_{2,u,var} \cdot 2 \cdot 100 \cdot K_{2,u,vol}$ (where $K_{2,u,vol}$ is in absolute terms).
Allowable values: Any real number.

- [number] `Caps2`: For each underlying u in the `Underlyings2` basket, the cap on the realised variance as a factor of the strike. For example, if the value in `Caps2` is 2.5, then the cap level will be $2.5^2 \times K_{2,u,\text{var}}$. For underlyings with no cap, set the value in `Caps2` to zero.
Allowable values: Any non-negative number.

- [number] `Floors2`: For each underlying u in the `Underlyings2` basket, the floor on the realised variance as a factor of the strike. For example, if the value in `Floors2` is 0.1, then the floor level will be $0.1^2 \times K_{2,u,\text{var}}$. For underlyings with no floor, set the value in `Floors2` to zero.
Allowable values: Any non-negative number.

- [number] `UpperBarrierLevels`: The agreed upper barrier price levels for each underlying in `Underlyings1`.
Allowable values: Any real number.

- [number] `LowerBarrierLevels`: The agreed lower barrier price levels for each underlying in `Underlyings1`.
Allowable values: Any real number.

- [bool] `CountBothObservations`: Whether the variance is accrued based on both the current and previous underlying prices (of each underlying in `Underlyings1`)

falling within the corridor (*True*) or only on the previous underlying price (*False*).
Allowable values: Boolean node, allowing *Y*, *N*, *1*, *0*, *true*, *false*, etc. The full set of allowable values is given in Table 29.

- [bool] `AccrualAdjustment`: Whether the strike will be scaled relative to the number of days that the underlying traded within the corridor. See, for example, $\widetilde{K}_{var}$ in section Corridor Variance Swap of the Product Description for Exotic Variance and Volatility Swaps.
- [bool] `DividendAdjustment`: Whether or not the underlying price is adjusted for dividend payments. This feature is not yet supported. Allowable values: Boolean node, allowing *Y*, *N*, *1*, *0*, *true*, *false*, etc. The full set of allowable values is given in Table 29.
- [event] `ValuationSchedule`: The schedule defining the (daily) observation period for the variance accrual.
Allowable values: See Section 2.3.4.
- [event] `SettlementDate`: The date on which the swap payoff is settled.
Allowable values: See `Date` in Table 13.
- [currency] `PayCcy`: The payment currency. For FX, where the underlying is provided in the form `FX-SOURCE-CCY1-CCY2` (see Table 21) this should be `CCY2`. If `CCY1` or the currency of the underlying (for EQ and COMM underlyings), this will result in a quanto payoff. Notice section "Payment Currency" in ore/Docs/ScriptedTrade.
Allowable values: See Table 15 for allowable currency codes.

KO Corridor Variance Dispersion Swap

This instrument is a variance dispersion swap with a double-barrier knock-out and a corridor feature.

With the corridor feature, variance is accrued only when the the price of the first underlying of each pair of underlyings (between the first and second basket) falls within the 'corridor'. This means that the number of underlyings in each basket should be the same, and each pair is determined based on the order that they are provided in the trade XML, as in the example below.

The traditional trade representation is as follows, using EQ underlyings in this example:

```
<Trade id="EQ_VarianceDispersionSwap_Corridor_KO">
  <TradeType>ScriptedTrade</TradeType>
  <Envelope>
    .....
  </Envelope>
  <KOCorridorVarianceDispersionSwapData>
    <LongShort type="longShort">Long</LongShort>
    <Weights type="number">
        .....
    </Weights>
    <Underlyings1 type="index">
      <Value>EQ-RIC:HSBA.L</Value>
      <Value>EQ-RIC:MSFT.OQ</Value>
      <Value>EQ-RIC:AMZN.O</Value>
```

```
    </Underlyings1>
    <Strikes1 type="number">
        .....
    </Strikes1>
    <Spreads1 type="number">
        .....
    </Spreads1>
    <Notionals1 type="number">
        .....
    </Notionals1>
    <Caps1 type="number">
        .....
    </Caps1>
    <Floors1 type="number">
        .....
    </Floors1>
    <Underlyings2 type="index">
      <Value>EQ-RIC:.STOXX50E</Value>
      <Value>EQ-RIC:.SPX</Value>
      <Value>EQ-RIC:.SPX</Value>
    </Underlyings2>
    <Strikes2 type="number">
        .....
    </Strikes2>
    <Spreads2 type="number">
        .....
    </Spreads2>
    <Notionals2 type="number">
        .....
    </Notionals2>
    <Caps2 type="number">
        .....
    </Caps2>
    <Floors2 type="number">
        .....
    </Floors2>
    <CorridorUpperBarrierLevels type="number">
        .....
    </CorridorUpperBarrierLevels>
    <CorridorLowerBarrierLevels type="number">
        .....
    </CorridorLowerBarrierLevels>
    <KOUpperBarrierLevels type="number">
        .....
    <KOUpperBarrierLevels/>
    <KOLowerBarrierLevels type="number">
        .....
    </KOLowerBarrierLevels>
    <CountBothObservations type="bool">true</CountBothObservations>
    <AccrualAdjustment type="bool">true</AccrualAdjustment>
    <DividendAdjustment type="bool">false</DividendAdjustment>
    <KnockOutSchedule type="event">
      <ScheduleData>
        .....
      </ScheduleData>
    </KnockOutSchedule>
    <VarianceAccrualStartDate type="event">2020-12-02</VarianceAccrualStartDate>
    <SettlementSchedule type="event">
      <DerivedSchedule>
        <BaseSchedule>KnockOutSchedule</BaseSchedule>
          .....
      </DerivedSchedule>
    </SettlementSchedule>
    <PayCcy type="currency">USD</PayCcy>
  </KOCorridorVarianceDispersionSwapData>
</Trade>
```

The KOCorridorVarianceDispersionSwap script referenced in the trade above is shown in listing 162

The meanings and allowable values for the `KOCorridorVarianceDispersionSwapData`

node below.

- [longShort] `LongShort`: Own party position in the swap. *Long* corresponds to selling volatility on the `Underlyings1` basket and buying volatility on `Underlyings2`.
  Allowable values: *Long, Short.*
- [number] `Weights`: List of weights applied to the final realised volatility of each underlying in the `Underlyings1` and `Underlyings2` baskets.
  Allowable values: Any positive number.
- [index] `Underlyings1`: The basket of underlyings whose volatility is bought in the *Long* position.
  Allowable values: For each underlying, see 2.3.29.
- [number] `Strikes1`: The volatility strike $K_{1,u,vol}$ of the variance swap for each underlying u in the `Underlyings1` basket, quoted in absolute terms. If the swap was struck in terms of variance, the square root of that variance should be used here.
  Allowable values: Any positive number, as a percentage expressed in decimal form.
- [number] `Spreads1`: Additional spread to the realised variance, for each underlying in the `Underlyings1` basket.
  Allowable values: Any real number.
- [number] `Notionals1`: For each underlying u in the `Underlyings1` basket, the vega notional amount. If the swap was struck in terms of variance notionals $N_{1,u,var}$, the corresponding vega notionals are given by $N_{1,u,vol} = N_{1,u,var} \cdot 2 \cdot 100 \cdot K_{1,u,vol}$ (where $K_{1,u,vol}$ is in absolute terms).
  Allowable values: Any real number.
- [number] `Caps1`: For each underlying u in the `Underlyings1` basket, the cap on the realised variance as a factor of the strike. For example, if the value in `Caps1` is 2.5, then the cap level will be $2.5^2 \times K_{1,u,\text{var}}$. For underlyings with no cap, set the value in `Caps1` to zero.
  Allowable values: Any non-negative number.
- [number] `Floors1`: For each underlying u in the `Underlyings1` basket, the floor on the realised variance as a factor of the strike. For example, if the value in `Floors1` is 0.1, then the floor level will be $0.1^2 \times K_{1,u,\text{var}}$. For underlyings with no floor, set the value in `Floors1` to zero.
  Allowable values: Any non-negative number.
- [index] `Underlyings2`: The basket of underlyings whose volatility is sold in the *Long* position.
  Allowable values: For each underlying, see 2.3.29.
- [number] `Strikes2`: The volatility strike $K_{2,u,vol}$ of the variance swap for each underlying u in the `Underlyings2` basket, quoted in absolute terms. If the swap was struck in terms of variance, the square root of that variance should be used here.

Allowable values: Any positive number, as a percentage expressed in decimal form.

- [number] `Spreads2`: Additional spread to the realised variance, for each underlying in the `Underlyings2` basket.
  Allowable values: Any real number.

- [number] `Notionals2`: For each underlying u in the `Underlyings2` basket, the vega notional amount. If the swap was struck in terms of variance notionals $N_{2,u,var}$, the corresponding vega notionals are given by $N_{2,u,vol} = N_{2,u,var} \cdot 2 \cdot 100 \cdot K_{2,u,vol}$ (where $K_{2,u,vol}$ is in absolute terms).
  Allowable values: Any real number.

- [number] `Caps2`: For each underlying u in the `Underlyings2` basket, the cap on the realised variance as a factor of the strike. For example, if the value in `Caps2` is 2.5, then the cap level will be $2.5^2 \times K_{2,u,\text{var}}$. For underlyings with no cap, set the value in `Caps2` to zero.
  Allowable values: Any non-negative number.

- [number] `Floors2`: For each underlying u in the `Underlyings2` basket, the floor on the realised variance as a factor of the strike. For example, if the value in `Floors2` is 0.1, then the floor level will be $0.1^2 \times K_{2,u,\text{var}}$. For underlyings with no floor, set the value in `Floors2` to zero.
  Allowable values: Any non-negative number.

- [number] `CorridorUpperBarrierLevels`: The agreed corridor upper barrier price levels for each underlying in `Underlyings1`.
  Allowable values: Any real number.

- [number] `CorridorLowerBarrierLevels`: The agreed corridor lower barrier price levels for each underlying in `Underlyings1`.
  Allowable values: Any real number.

- [number] `KOUpperBarrierLevels`: The agreed knock-out upper barrier price levels for each underlying in `Underlyings1`.
  Allowable values: Any real number.

- [number] `KOLowerBarrierLevels`: The agreed knock-out lower barrier price levels for each underlying in `Underlyings1`.
  Allowable values: Any real number.

- [bool] `CountBothObservations`: Whether the variance is accrued based on both the current and previous underlying prices (of each underlying in `Underlyings1`) falling within the corridor (*True*) or only on the previous underlying price (*False*).
  Allowable values: Boolean node, allowing *Y*, *N*, *1*, *0*, *true*, *false*, etc. The full set of allowable values is given in Table 29.

- [bool] `AccrualAdjustment`: Whether the strike will be scaled relative to the number of days that the underlying traded within the corridor. See, for example, $\widetilde{K}_{var}$ in section Corridor Variance Swap of the Product Description for Exotic Variance and Volatility Swaps.

- [bool] `DividendAdjustment`: Whether or not the underlying price is adjusted for

dividend payments. This feature is not yet supported. Allowable values: Boolean node, allowing *Y*, *N*, *1*, *0*, *true*, *false*, etc. The full set of allowable values is given in Table 29.

- [event] `KnockOutSchedule`: The schedule defining the (daily) knock-out observation period.
  Allowable values: See Section 2.3.4.
- [event] `VarianceAccrualStartDate`: The date on which variance observation/accrual starts. The last variance observation date is the same as the last date in the `KnockOutSchedule`.
  Allowable values: See `Date` in Table 13.
- [event] `SettlementSchedule`: The settlement dates derived from the `KnockOutSchedule` using a settlement lag to reflect settlement after a knock-out event, or after the final variance observation date if no knock-out occurs..
  Allowable values: See section 2.3.4 Schedule Data and Dates, or DerivedSchedule (see (see the scripted trade documentation in ore/Docs/ScriptedTrade).
- [currency] `PayCcy`: The payment currency. For FX, where the underlying is provided in the form `FX-SOURCE-CCY1-CCY2` (see Table 21) this should be `CCY2`. If `CCY1` or the currency of the underlying (for EQ and COMM underlyings), this will result in a quanto payoff. Notice section "Payment Currency" in ore/Docs/ScriptedTrade.
  Allowable values: See Table 15 for allowable currency codes.

Gamma Swap

The traditional trade representation is as follows, using EQ underlyings in this example:

```
<Trade id="EQ_GammaSwap">
  <TradeType>ScriptedTrade</TradeType>
  <Envelope>
   .....
  </Envelope>
    <GammaSwapData>
  <LongShort type="longShort">Long</LongShort>
  <Strike type="number">0.19</Strike>
  <Notional type="number">1380</Notional>
  <SettlementDate type="event">2021-09-25</SettlementDate>
  <Underlying type="index">EQ-RIC:.SPX</Underlying>
  <ValuationSchedule type="event">
    <ScheduleData>
      <Rules>
    <StartDate>2020-11-26</StartDate>
    <EndDate>2021-09-18</EndDate>
    <Tenor>1D</Tenor>
    <Convention>Following</Convention>
    <TermConvention>Following</TermConvention>
    <Calendar>USA</Calendar>
    <Rule>Forward</Rule>
       </Rules>
     </ScheduleData>
  </ValuationSchedule>
  <PayCcy type="currency">USD</PayCcy>
  </GammaSwapData>
</Trade>
```

The GammaSwap script referenced in the trade above is shown in listing 163

The meanings and allowable values for the `GammaSwapData` node below.

- [longShort] `LongShort`: Own party position in the swap. *Long* corresponds to paying out on the fixed/strike variance (volatility) and receiving on the floating/realised variance (volatility). In other words, a long position has positive value if the realised variance (volatility) exceeds the variance (volatility) strike.
  Allowable values: *Long, Short*.
- [number] `Strike`: The volatility strike K of the underlying quoted in absolute terms. If the swap was struck in terms of variance, the square roots of the variances should be used here.
  Allowable values: Any non-negative number, as a percentage expressed in decimal form.
- [number] `Notional`: The vega notional amount. If the swap was struck in terms of a variance notional N_{var}, the corresponding vega notional is given by $N_{vol} = N_{var} \cdot 2 \cdot 100 \cdot K$ (where K is in absolute terms).
  Allowable values: Any non-negative number.
- [event] `SettlementDate`: The date on which the swap payoff is settled.
  Allowable values: See `Date` in Table 13.
- [index] `Underlying`: Underlying index.
  Allowable values: See ore/Docs/ScriptedTrade's Index Section for allowable values.
- [event] `ValuationSchedule`: The base schedule which defines the start dates of the variance accrual schedule. This node is also used to derive the variance accrual schedule. Note that the end date (or last date) of this schedule must be the final period end date of the instrument.
  Allowable values: See Section 2.3.4.
- [currency] `PayCcy`: The payment currency. For FX, where the underlying is provided in the form `FX-SOURCE-CCY1-CCY2` (see Table 21) this should be `CCY2`. If `CCY1` or the currency of the underlying (for EQ and COMM underlyings), this will result in a quanto payoff. Notice section "Payment Currency" in ore/Docs/ScriptedTrade.
  Allowable values: See Table 15 for allowable currency codes.

Basket Variance Swap

The `FxBasketVarianceSwap` and `EquityBasketVarianceSwap` `CommodityBasketVarianceSwap` trade types have trade data containers (respectively):

- `FxBasketVarianceSwapData`
- `EquityBasketVarianceSwapData`
- `CommodityBasketVarianceSwapData`

Listing 164 shows the structure of example trades for an Equity underlying.

Listing 164: EquityBasketVarianceSwap data

```
<Trade id="Equity_VarianceSwap_Basket">
  <TradeType>EquityBasketVarianceSwap</TradeType>
  <Envelope>
    .......
  </Envelope>
  <EquityBasketVarianceSwapData>
    <LongShort>Long</LongShort>
    <Strike>0.31613</Strike>
    <Notional>1340.0</Notional>
    <Underlyings>
      <Underlying>
        <Type>Equity</Type>
        <Name>RIC:.SPX</Name>
        <Weight>0.75</Weight>
      </Underlying>
      <Underlying>
        <Type>Equity</Type>
        <Name>RIC:.STOXX50E</Name>
        <Weight>0.25</Weight>
      </Underlying>
    </Underlyings>
    <SquaredPayoff>true</SquaredPayoff>
    <ValuationSchedule>
        <Rules>
        .......
        </Rules>
    </ValuationSchedule>
    <Cap>2.5</Cap>
    <SettlementDate>2022-06-29</SettlementDate>
    <Currency>USD</Currency>
  </EquityBasketVarianceSwapData>
</Trade>
```

The payout formula is the same as for a standard capped/floored variance swap, with the realised variance calculated as follows:

$$RealisedVariance = \frac{252}{D}\sum_{i=1}^{D}\left[\sum_{n=1}^{N} w_n \ln\left(\frac{X_{i,n}}{X_{i-1,n}}\right)\right]^2,$$

for D variance accrual/valuation dates over the life of the trade, where

- $X_{i,n}$ denotes the fixing for the n-th underlying at t_i.
- w_n is the basket weight for the n-th underlying.

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

- `LongShort`: Own party position.
  Allowable values: *Long*, *Short*
- `Strike`: The volatility strike K_{vol} of the variance swap quoted in absolute terms. If the swap was struck in terms of variance, the square root of that variance should be used here.
  Allowable values: Any non-negative number, as a percentage expressed in decimal form.
- `Notional`: The vega notional amount. If the swap was struck in terms of a variance notional N_{var}, the corresponding vega notional is given by

$N_{vol} = N_{var} \cdot 2 \cdot 100 \cdot K_{vol}$ (where K_{vol} is in absolute terms).
Allowable values: Any non-negative number.

- `Underlyings`: The basket of underlyings.
  For `FxBasketVarianceSwap`, `Type` is set to *FX* and `Name` is a string of the form *SOURCE-CCY1-CCY2* where *CCY1* is the foreign currency, *CCY2* is the domestic currency, and *SOURCE* is the fixing source, see Table 21. (section Data Node / Index) for allowable values.

  For `EquityBasketVarianceSwap`, `Type` is set to *Equity* and `Name` and other fields are as outlined in 2.3.29.

  For `CommodityBasketVarianceSwap`, `Type` is set to *Commodity* and `Name` is an identifier of the commodity as outlined in 2.3.29 and in Table 25

  Allowable values: For each underlying, an `Underlying` node as outlined in 2.3.29. All underlyings must be from the same asset class.

- `SquaredPayoff` [Optional]: Flag indicating whether the trade is a variance swap (*True*) or a volatility swap (*False*).
  Allowable values: Boolean node, allowing *Y*, *N*, *1*, *0*, *true*, *false*, etc. Defaults to *False* if left blank or omitted. The full set of allowable values is given in Table 29.

- `ValuationSchedule`: The schedule defining the (daily) observation period for the variance accrual.
  Allowable values: See Section 2.3.4.

- `Cap` [Optional]: The cap on the realised variance, as a factor of the corresponding strike. For example, if `Cap` is 2.5, then the cap level will be $2.5^2 \times \hat{K}_{var}$ for the basket realised variance. For trades with no cap, set `Cap` to zero.
  Allowable values: Any non-negative number. Defaults to zero if left blank or omitted.

- `Floor` [Optional]: The floor on the realised variance, as a factor of the corresponding strike. For example, if `Floor` is 0.1, then the floor level will be $0.1^2 \times \hat{K}_{var}$ for the basket realised variance. For trades with no floor, set `Floor` to zero.
  Allowable values: Any non-negative number. Defaults to zero if left blank or omitted.

- `SettlementDate`: The date on which the swap payoff is settled.
  Allowable values: See `Date` in Table 13.

- `Currency`: The settlement currency.
  Allowable values: See Table 15 `Currency`.

Basket variance swaps can alternatively be represented as *scripted trades*, refer to the scripted trade documentation in ore/Docs/ScriptedTrade for an introduction.

```
<Trade id="EQ_VarianceSwap_Basket">
  <TradeType>ScriptedTrade</TradeType>
  <Envelope>
    ......
  </Envelope>
  <BasketVarianceSwapData>
    <LongShort type="longShort">Long</LongShort>
    <Underlyings type="index">
```

```
      <Value>EQ-RIC:.STOXX50E</Value>
      <Value>EQ-RIC:.SPX</Value>
    </Underlyings>
    <Weights type="number">
      <Value>0.25</Value>
      <Value>0.75</Value>
    </Weights>
    <Strike type="number">0.31613</Strike>
    <Notional type="number">1339.0</Notional>
    <SquaredPayoff type="bool">true</SquaredPayoff>
    <ValuationSchedule type="event">
      <ScheduleData>
        <Rules>
         ......
        </Rules>
      </ScheduleData>
    </ValuationSchedule>
    <Cap type="number">2.5</Cap>
    <Floor type="number">0</Floor>
    <SettlementDate type="event">2022-06-29</SettlementDate>
    <PayCcy type="currency">USD</PayCcy>
  </BasketVarianceSwapData>
</Trade>
```

The PairwiseVarianceSwap script referenced in the trade above is shown in listing 165

The meanings and allowable values for the `BasketVarianceSwapData` node below.

- [longShort] `LongShort`: Own party position in the swap. *Long* corresponds to paying out on the fixed/strike variance (volatility) and receiving on the floating/realised variance (volatility). In other words, a long position has positive value if the realised variance (volatility) exceeds the variance (volatility) strike.
  Allowable values: *Long, Short*.

- [index] `Underlyings`: List of underlying indices.
  Allowable values: See ore/Docs/ScriptedTrade's Index Section for allowable values.

- [number] `Strike`: The volatility strike K_{vol} of the variance swap quoted in absolute terms. If the swap was struck in terms of variance, the square root of that variance should be used here.
  Allowable values: Any non-negative number, as a percentage expressed in decimal form.

- [number] `Notional`: The vega notional amount. If the swap was struck in terms of a variance notional N_{var}, the corresponding vega notional is given by $N_{vol} = N_{var} \cdot 2 \cdot 100 \cdot K_{vol}$ (where K_{vol} is in absolute terms).
  Allowable values: Any non-negative number.

- `SquaredPayoff`: Flag indicating whether the trade is a variance swap (*True*) or a volatility swap (*False*).
  Allowable values: Boolean node, allowing *Y*, *N*, *1*, *0*, *true*, *false*, etc. Defaults to *False* if left blank or omitted. The full set of allowable values is given in Table 29.

- [event] `ValuationSchedule`: The schedule defining the (daily) observation period for the variance accrual.
  Allowable values: See Section 2.3.4.

- [number] `Cap`: The cap on the realised variance, as a factor of the corresponding strike. For example, if `Cap` is 2.5, then the cap level will be $2.5^2 \times \hat{K}_{var}$ for the

basket realised variance. For trades with no cap, set `Cap` to zero.
Allowable values: Any non-negative number.

- [number] `Floor`: The floor on the realised variance, as a factor of the corresponding strike. For example, if `Floor` is 0.1, then the floor level will be $0.1^2 \times \hat{K}_{var}$ for the basket realised variance. For trades with no floor, set `Floor` to zero.
  Allowable values: Any non-negative number.
- [event] `SettlementDate`: The date on which the swap payoff is settled.
  Allowable values: See `Date` in Table 13.
- [currency] `PayCcy`: The settlement currency.
  Allowable values: See Table 15 for allowable currency codes.

2.2.84 Generic Scripted Products

The products in sections 2.2.70 to 2.2.80 are internally represented as *Scripted Trades*, but "wrapped" such that their input XML format still looks like "classic" ORE XML.

With 2.2.82 and 2.2.83 we have seen two examples of the generic Scripted Trade input format.

The Scripted Trade module allows flexible definition of new payoffs across five of the six asset classes covered in ORE, just by way of defining the payoff script. The payoff script can be embedded into the trade XML or can be placed into a separate script library.

Refer to the stand-alone Scripted Trade documentation in ore/Docs/ScriptedTrade or section 4 for an introduction.

2.2.85 Flexi Swap

Payoff

A Flexi Swap is an amortizing swap in which one party has the option to further reduce the notional in each period to each value between the current notional and a specified lower bound for that period.

Flexi Swaps are priced in ORE following the Replication Approach described in (F. Jamshidian, 2005). The replicating basket of Bermudan Swaptions is priced using the method as described under Product Type "Bermudan Swaption", but using one global calibration for all swaptions derived from the flexi swap structure. Prepayment option types "ReductionByAbsoluteAmount" and "ReductionUpToAbsoluteAmount" are both treated approximately by generating a schedule of deterministic, lower notional bounds from assuming that all earlier prepayment options were executed.

Input

The `FlexiSwapData` node is the trade data container for trade type Flexi Swap. A Flexi Swap is a two-legged swap with optional and customisable pre-payments. Flexi Swaps are typically used for representing swaps linked to Asset Backed Securities with flexible amortisation. A Flexi Swap must have two legs, one fixed and one floating. The floating leg must have a pay frequency that is a multiple of the fixed leg frequency

and corresponding floating and fixed leg periods must have the same notional. The legs typically have an amortising notional and are represented by `LegData` trade component sub-nodes, described in section 2.3.3. The `FlexiSwapData` node also contains a `OptionLongShort` node indicating the holder of the prepayment option and a node describing the optional prepayments, see below.

An example structure of a `FlexiSwapData` node is shown in Listing 166. In this case the optional pre-payments are given by a subnode `LowerNotionalBounds` meaning that the notional of the swap can be reduced to any value between the given lower bound and the original notional in each fixed leg period.

Listing 166: Flexi Swap data

```
<FlexiSwapData>
  <LowerNotionalBounds>
        <Notional>451389557.145667</Notional>
        <Notional>427876791.621303</Notional>
        <Notional>404435982.369285</Notional>
        <Notional>379353200.32956</Notional>
        ...
  </LowerNotionalBounds>
  <OptionLongShort>Short</OptionLongShort>
  <LegData>
    <LegType>Fixed</LegType>
    ...
  </LegData>
  <LegData>
    <LegType>Floating</LegType>
    ...
  </LegData>
</FlexiSwapData>
```

Alternatively the optional pre-payments can be described by a subnode `NotionalDecreases` which is more general than the description via LowerNotionalBounds (using the reduction type RedutionToLowerBound, see below for more details on this), see Listing 167 for an example.

Listing 167: Flexi Swap data

```
<FlexiSwapData>
  <Prepayment>
    <NoticePeriod>5D</NoticePeriod>
    <NoticeCalendar>TARGET</NoticeCalendar>
    <NoticeConvention>F</NoticePeriod>
    <PrepaymentOptions>
      <PrepaymentOption>
        <ExerciseDate>2015-02-01</ExerciseDate>
        <Type>ReductionUpToLowerBound</Type>
        <Value>404435982.369285</Value>
      </PrepaymentOption>
      <PrepaymentOption>
        <ExerciseDate>2016-02-01</ExerciseDate>
        <Type>ReductionByAbsoluteAmount</Type>
        <Value>100000.0</Value>
      </PrepaymentOption>
      <PrepaymentOption>
        <ExerciseDate>2017-02-01</ExerciseDate>
        <Type>ReductionUpToAbsoluteAmount</Type>
        <Value>50000.0</Value>
      </PrepaymentOption>
    <PrepaymentOptions>
  </Prepayment>
  <OptionLongShort>Short</OptionLongShort>
  <LegData>
    <LegType>Fixed</LegType>
    ...
  </LegData>
  <LegData>
    <LegType>Floating</LegType>
    ...
  </LegData>
</FlexiSwapData>
```

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

- OptionLongShort: Specifies which party has the right to pre-pay the notional down to the lower notional bound. *Short* means that for pricing purposes pre-payments are assumed to be done in such a way to maximise the value of the Flexi Swap for the "other" counterparty, *Long* means that the Flexi Swap value is maximised from "our" point of view.

  Allowable values: *Long* or *Short*

- LegData: This is a trade component sub-node outlined in section 2.3.3. A Flexi Swap must have two `LegData` nodes and the LegType element must be set to *Floating* on one leg and *Fixed* on the other. The two legs must have the same `Currency`. The float leg pay frequency must be a multiple of the fixed leg frequency.

The optional prepayments are described by either a `LowerNotionalBounds` node or a `Prepyment` node.

In case the optional prepayments are described by a `LowerNotionalBounds` node, the minimum level to which the notional can be amortised down to must be given as a notional schedule. The schedule can be specified as described in 2.3.3, i.e. using a sequence of `Notional` subnodes or using the `startDate` attribute to specify notional changes. The given schedule must be given for the fixed leg periods since the notional can be decreased for each whole fixed leg period and the corresponding floating leg periods (remember that the floating leg frequency must be a multiple of the fixed leg frequency). Each lower notional bound child element can take a positive real number that cannot exceed the notional amount of the corresponding coupon period on either leg and (from the second fixed coupon period on) the lower notional bound of the previous coupon period.

In case the optional prepayments are described by a `Prepayment` node, the the single exercise opportunities are described by a `PrepaymentOptions` subnode that contains one or several `PerpaymentOption` subnodes, each of which comprises the following elements:

- ExerciseDate: The date on which the notional can be decreased.
- Type: The type of the allowed notional reduction. The allowable types are
  - ReductionUpTpLowerBound: The notional can be reduced to any value between the current notional and the lower bound given in the Value node.
  - ReductionByAbsoluteAmount: The notional can be reduced by an absolute amount given in the Value node. If this value is greater than the current notional, the reduction amount is equal to the current notional.
  - ReductionUpToAbsoluteAmount: The notional can be reduced by any value between zero and a given absolute amount (given in the Value node).
- Value: The value that together with the type describes the amount by which the notional can be decreased.

In addition the `Prepayment` node contains the following optional subnodes describing the conventions for deriving the option notice date from the exercise date:

- NoticePeriod [Optional]: The notice period defining the date (relative to the exercise date) on which the exercise decision has to be taken. If not given the notice period defaults to 0D, i.e. the notice date is identical to the exercise date.
- NoticeCalendar [Optional]: The calendar used to compute the notice date from the exercise date. If not given defaults to the null calendar (no holidays, weekends are no holidays either).
- NoticeConvention [Optional]: The convention used to compute the notice date from the exercise date. Defaults to Unadjusted if not given.

2.2.86 Balance Guaranteed Swap (BGS)

Payoff

A Balance Guaranteed Swap is similar to an amortizing interest rate swap, but the notional amortization matches actual prepayments of a Reference Security which can be either a tranche or a reference pool of assets, or securitized interest backed by a

pool of assets. The BGS differs from an amortizing swap in that the notional amortizations are uncertain.

Input

BGS are priced in ORE using an auxiliary Flexi Swap as a proxy. The amortization schedule of the Flexi Swap is set up as the notional schedule of the BGS assuming a zero CPR (Conditional Prepayment Rate). The lower notional bound of the Flexi Swap is constructed assuming a MaxCPR (Maximum Conditional Prepayment Rate) which is dependent on the Reference Security. The MaxCPR is estimated on the basis of the current CPR, historical CPRs and / or expert judgement as to provide a (hypothetical) sufficiently realistic hedge for the BGS. The option holder in the Flexi Swap is the payer of the structured leg (i.e. the leg replicating the payments of the reference security) in the BGS.

The `BalanceGuaranteedSwapData` node is the trade data container for trade type *BalanceGuaranteedSwap*. A BGS must have two legs, one fixed and one floating. Each leg typically has an amortising notional and is represented by a `LegData` trade component sub-node, described in section 2.3.3. The `BalanceGuaranteedSwapData` node also contains a `ReferenceSecurity` sub-node specifying the Asset Backed Security to which the notional schedule of the BGS is linked. An example structure of a `BalanceGuaranteedSwapData` node is shown in Listing 168.

Listing 168: Balance Guaranteed Swap data

```
<BalanceGuaranteedSwapData>
  <ReferenceSecurity>ISIN:XS0983610930</ReferenceSecurity>
  <Tranches>
    <Tranche>
      <Description>Class A</Description>
      <SecurityId>ISIN:XS0983610930</SecurityId>
      <Seniority>1</Seniority>
      <Notionals>
      ...
      </Notionals>
    </Tranche>
    <Tranche>
      <Description>Class B</Description>
      <SecurityId>ISIN:XS0983610931</SecurityId>
      <Seniority>2</Seniority>
      <Notionals>
      ...
      </Notionals>
    </Tranche>
    <ScheduleData>
    ...
    </ScheduleData>
  </Tranches>
  <LegData>
        <LegType>Fixed</LegType>
         ...
  </LegData>
  <LegData>
        <LegType>Floating</LegType>
         ...
  </LegData>
<BalanceGuaranteedSwapData>
```

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

- ReferenceSecurity: The ISIN of the Asset Backed Security tranche to which the BGS is linked.

  Allowable values: The prefix `ISIN:` followed by an ISIN code for the Reference Security.

- Tranches: A description of the Asset Backed Security tranche notionals. Each Tranche is identified by a `SecurityId` and an optional `Description`. Each Tranche has a `Seniority` given as a positive integer value where lower values mean higher seniority, i.e. 1 is the most senior tranche (e.g. "class A") followed by 2 (e.g. "class B") etc. The notionals are given in a sub-node `Notionals` as described in section 2.3.3 w.r.t. a schedule given in `ScheduleData` which is shared across all tranches. There must be exactly one tranche with a security id matching the reference security.

- LegData: This is a trade component sub-node outlined in section 2.3.3. A BGS must have two `LegData` nodes and the LegType element must be set to *Floating*

on one leg and *Fixed* on the other. The two legs must have the same `Currency`.

The notionals of the swap and the referenced tranche must be consistent. Furthermore, notionals for periods with a start date in the past must be given with their actual value, i.e. including actual prepayments that were made in the previous periods. Notionals for periods with a start dats in the future on the other hand must be given assuming a zero conditional prepayment rate. For the latter periods a prepayment model is used to generate suitable notional schedules when pricing the swap. The prepayment model assumes that tranches with higher seniority are amortised first, i.e. in the example here the class A tranche is amortised before the class B tranche.

2.3 Trade Components

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

An example of a `SwapData` trade data container, including two `LegData` trade components which in turn include further trade components such as `FixedLegData`, `ScheduleData` and `FloatingLegData` is shown in Listing 169.

Listing 169: Trade Components Example

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

Descriptions of all trade components supported in ORE follow below.

2.3.1 Option Data

This trade component node is used within the `SwaptionData` and `FXOptionData` trade data containers. It contains the `ExerciseDates` sub-node which includes `ExerciseDate` child elements. An example structure of the `OptionData` trade component node is shown in Listing 170.

Listing 170: Option data

```
<OptionData>
  <LongShort>Long</LongShort>
  <OptionType>Call</OptionType>
  <Style>Bermudan</Style>
  <NoticePeriod>5D</NoticePeriod>
  <NoticeCalendar>TARGET</NoticeCalendar>
  <NoticeConvention>F</NoticePeriod>
  <Settlement>Cash</Settlement>
  <SettlementMethod>CollateralizedCashPrice</SettlementMethod>
  <MidCouponExercise>false</MidCouponExercise>
  <PayOffAtExpiry>true</PayOffAtExpiry>
  <ExerciseFees>
    <ExerciseFee type="Percentage">0.0020</ExerciseFee>
    <ExerciseFee type="Absolute" startDate="2020-04-20">25000</ExerciseFee>
  </ExerciseFees>
  <ExerciseFeeSettlementPeriod>2D</ExerciseFeeSettlementPeriod>
  <ExerciseFeeSettlementConvention>F</ExerciseFeeSettlementConvention>
  <ExerciseFeeSettlementCalendar>TARGET</ExerciseFeeSettlementCalendar>
  <ExerciseDates>
    <ExerciseDate>2019-04-20</ExerciseDate>
    <ExerciseDate>2020-04-20</ExerciseDate>
  </ExerciseDates>
  <!-- Alternative format for exercise dates using Schedule format -->
  <ExerciseSchedule>
    <Rules>
      <StartDate>2019-04-20</StartDate>
      <EndDate>2024-04-20</EndDate>
      <Tenor>3M</Tenor>
    </Rules>
  </ExerciseSchedule>
  <Premiums>
    <Premium>
      <Amount>100000</Amount>
      <Currency>EUR</Currency>
      <PayDate>2018-05-07</PayDate>
    </Premium>
  </Premiums>
  <AutomaticExercise>...</AutomaticExercise>
  <ExerciseData>
    <Date>...</Date>
    <Price>...</Price>
  </ExerciseData>
  <PaymentData>...</PaymentData>
</OptionData>
```

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

- LongShort: Specifies whether the option position is *long* or *short*. Note that for Swaptions, Callable Swaps, and Index CDS Options setting `LongShort` to *short* makes the `Payer` indicator on the underlying Swap / Index CDS to be set from the perspective of the Counterparty.

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

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

  Allowable values: *Call* or *Put*

  The meaning of Call and Put values depend on the trade type and asset class of the option, see Table 7.

Asset Class and Trade Type	Call / Put Specifications
Equity/ Commodity/Bond Option	*Call*: The right to buy the underlying equity/commodity/bond at the strike price. *Put*: The right to sell the underlying equity/commodity/bond at the strike price.
IR Swaption, CallableSwap, Commodity Swaption	*Call/Put* values are ignored, and the OptionType field is optional. Payer/Receiver swaption is determined by the `Payer` fields in the Leg Data nodes of the underlying swap.
FX Options (all variants, except Touch, Digital, Asian)	*Call*: Bought and Sold currencies/amounts stay as determined in the trade data node. *Put*: Bought and Sold currencies/amounts are switched compared to the trade data node. Note that barriers are not switched / unaffected.
Index CDS Option	*Call/Put* values are ignored, and the OptionType field is optional. The `Payer` field in the underlying Index CDS leg determines if the option is to buy or sell protection.
Asian FX Options	*Call*: The right to buy/receive the underlying currency at the strike price. *Put*: The right to sell/pay the underlying currency at the strike price.
Digital FX Options	*Call*: The digital payout will occur if the fx rate at the expiry date is above the given strike, *Put*: The digital payout will occur if the fx rate at the expiry date is below the given strike.
FX Single Touch Options	*Call/Put* values are ignored, and are instead inferred from the BarrierData type, and the OptionType field is optional.
FX Double Touch Options	*Call/Put* values are ignored, and and the OptionType field is optional.
Ascot	*Call* has payout: $\max(0, convertiblePrice - Strike)$ *Put* has payout: $\max(0, Strike - convertiblePrice)$

Table 7: Specification of Option Type Call / Put

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

Allowable values:

  - *Accumulator, Decumulator* (applies to trade types EquityAccumulator, FxAccumulator, CommodityAccumulator only)
  - *TargetFull, TargetExact, TargetTruncated* (applies to trade types EquityTaRF, FxTaRF, CommodityTaRF only)
  - *BestOfAssetOrCash, WorstOfAssetOrCash, MaxRainbow, MinRainbow* (applies to trade types EquityRainbowOption, FxRainbowOption, CommodityRainbowOption only)
  - *Vanilla, Asian, AverageStrike, LookbackCall, LookbackPut* (applies to trade types EquityBasketOption, FxBasketOption, CommodityBasketOption only)
  - *Asian* (applies to trade types EquityAsianOption, FxAsianOption only)
  - *Vanilla, AssetOrNothing, CashOrNothing* (applies to trade type FxGenericBarrierOption, EquityGenericBarrierOption, CommodityGenericBarrierOption)

- Style: The exercise style of the option.

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

  Note that trade types IR Swaption and CallableSwap can have all three styles: *European*, *Bermudan*, or *American*.

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

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

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

  Index CDS Options must have style *European*.

  Ascots must have style *American*.

- PayoffType2 [Optional]: Subtype for payoff of exotic options. Only applicable to specific trade types as indicated in parantheses:

  Allowable values:

  - *Arithmetic, Geometric* (applies to trade types EquityAsianOption, FxAsianOption only, if not given it defaults to Arithmetic)

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

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

- NoticeConvention [Optional]: The convention used to compute the notice date from the exercise date. Defaults to Unadjusted if not given.
- Settlement: Delivery type. Note that Settlement is not required for Asian options.

  Allowable values: *Cash* or *Physical*
- SettlementMethod [Optional]: Specifies the method to calculate the settlement amount for Swaptions and CallableSwaps.

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

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

  PhysicalOTC = OTC traded swaptions with physical settlement
  PhysicalCleared = Cleared swaptions with physical settlement
  CollateralizedCashPrice = Cash settled swaptions with settlement price calculation using zero coupon curve discounting
  ParYieldCurve = Cash settled swaptions with settlement price calculation using par yield discounting [5] [6]
- MidCouponExercise [Optional]: Relevant for Swaptions and CallableSwaps. If *false*, the exercise-into underlying comprises all coupons with accrual start date greater or equal to notification date. I.e. one exercises into the next coupon, not the current one.
  If *true*, the exercise-into underlying comprises all coupons with accrual end date greater than the effective exercise date which is computed from the notification date by adding the notice period. The accrual paid for such coupons on exercise is calculated from the effective exercise date to the accrual end date (short coupon).

  Allowable values: *true*, *false*. If omitted, defaults to *false* for European and Bermudan swaptions/callableswaps and *true* for American swaptions/callableswaps.
- PayOffAtExpiry [Optional]: Relevant for options with early exercise, i.e. the exercise occurs before expiry; *true* indicates payoff at expiry, whereas *false* indicates payoff at exercise. Defaults to *true* if left blank or omitted.

  Allowable values: *true*, *false*.

  Note that for `IndexCreditDefaultSwapOption` PayOffAtExpiry must be set to *false* as only payoff at exercise is supported.
- Premiums [Optional]: Option premium amounts paid by the option buyer to the option seller.

  Allowable values: See section 2.3.2

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

- ExerciseDates: This node contains child elements of type `ExerciseDate` or an `ExerciseSchedule` node.

  Options of style *European* require a single exercise date expressed by one single `ExerciseDate` child element.

  American style options must have exactly two `ExerciseDate` child elements representing the start and end of the American exercise period.

  Bermudan style options must have two or more `ExerciseDate` child elements. One can alternatively use `ExerciseSchedule` to specify the option exercise dates for *Bermudan* style options.

- ExerciseSchedule [Optional]: This node can be provided instead of `ExerciseDates` and should be specified in the same format as a Schedule (see Section 2.3.4), e.g. for a list of Bermudan exercise dates.

- ExerciseFees [Optional]: This node contains child elements of type ExerciseFee. Similar to a list of notionals (see 2.3.3) the fees can be given either

  – as a list where each entry corresponds to an exercise date and the last entry is used for all remaining exercise dates if there are more exercise dates than exercise fee entries, or

  – using the `startDate` attribute to specify a change in a fee from a certain day on (w.r.t. the exercise date schedule)

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

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

  Only supported for Swaptions and Callable Swaps currently.

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

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

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

  Allowable values: See Table 17 Calendar.

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

  Allowable values: See Table 14 Roll Convention.

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

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

  Allowable values: A boolean value given in Table 29. If not provided, the default value is *false*.

- ExerciseData [Optional]: Currently, this node is only used for vanilla European cash settled FX, equity and commodity options where *Automatic Exercise* is not applicable. It has the structure shown in Listing 170 i.e. a child `Date` and `Price` node. It is used to supply the price at which an option was exercised and the date of exercise. For a European option, the supplied date clearly has to match the single option `ExerciseDate`. It is needed where the cash settlement date is after the `ExerciseDate`. If this node is not supplied, and the `ExerciseDate` is in the past relative to the valuation date, the option is assumed to have expired unexercised.

  Allowable values: The `Date` node should be a valid date as outlined in Table 13 and the `Price` node should be a valid price as a real number.

- PaymentData [Optional]: This node is used to supply the date on which the option is cash settled if it is exercised. There are two methods in which this data may be supplied:

  1. The first method is an explicit list of dates as shown in Listing 171. The `Date` node should be a valid date as outlined in Table 13. Obviously, for European options, there should be exactly one date supplied.

  2. The second method is a set of rules that are used to generate the settlement date relative to either the exercise date of the option or the expiry date of the option. The structure of the `PaymentData` node in this case is given in Listing 172. The optional `RelativeTo` node must be either `Expiry` or `Exercise`. If it is `Expiry`, the expiry date is taken as the base date from which the rules are applied. If it is `Exercise`, the exercise date is taken as the base date from which the rules are applied. These two dates are the same in the case of a European option. If not provided, `Expiry` is assumed. The `Lag` node is a non-negative integer giving the number of days from the base date to the cash settlement date. The `Calendar` gives the business day calendar for the cash settlement date and should be a valid calendar code as outlined in Table 17. The `Convention` gives the roll convention for the cash settlement date and should be a valid roll convention as outlined in Table 14.

Listing 171: Dates based `PaymentData`

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

Listing 172: Rules based `PaymentData`

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

2.3.2 Premiums

The `Premiums` node holds data of one or more premiums to be paid. It is used in different trade types, notably in caps / floors (see section 2.2.3) and more generally in the option data component (see section 2.3.1). Listing 173 shows an example for a Premiums data block representing two premiums.

Listing 173: Premiums Node

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

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

- Amount: Option premium amounts paid by the option buyer to the option seller. A positive amount is considered to be paid by the option holder to the option seller and thus results in a negative contribution to the NPV of a long option.
  Allowable values: arbitrary number
- Currency: Currency of the premium to be paid
  Allowable values: See Table 15 `Currency`.
- PayDate: Date of the premium payment.
  Allowable values: See `Date` in Table 13.

We support a deprecated schema to represent a single premium as shown in listing 174 for backwards compatibility. The 3 nodes PremiumAmount, PremiumCurrency, PremiumPayDate can be used on the same level as the new Premiums node to represent a single premium payment. The deprecated and new schema may not be mixed.

Listing 174: Deprecated Single Premium Representation

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

2.3.3 Leg Data and Notionals

The `LegData` trade component node is used within the `CapFloorData`, `SwapData` , `SwaptionData` and `EquitySwapData` trade data containers. It contains a `ScheduleData` trade component sub-node, and a sub-node that depends on the value of the `LegType` element, e.g.: `FixedLegData` for `LegType` *Fixed* or `FloatingLegData` for `LegType` *Floating*. The `LegData` node also includes a `Notionals` sub-node with `Notional` child elements described below. An example structure of a `LegData` node of `LegType` *Floating* is shown in Listing 175.

Listing 175: Leg data

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

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

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

  Allowable values: *Fixed, Floating, Cashflow, CMS, CMB, DigitalCMS, DurationAdjustedCMS, CMSSpread, DigitalCMSSpread, Equity, CPI, YY, ZeroCouponFixed, FormulaBased, CommodityFloating, CommodityFixed, EquityMargin*

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

Allowable values: *true, false*

- Currency: The currency of the leg.

  Allowable values: See Table 15 `Currency`. When `LegType` is *Equity*, Minor Currencies in Table 15 are also allowable.

- PaymentCalendar [Optional]: The payment calendar of the leg coupons. The `PaymentCalendar` is used in conjunction with the `PaymentConvention`, `PaymentLag` and `NotionalPaymentLag` to determine the payments dates, unless the `PaymentDates` node is used which defines the payment dates explicitly.

  Allowable values: See Table 17 `Calendar`. If left blank or omitted, defaults to the calendar in the `ScheduleData` node, unless `LegType` is *Floating* and `Index` is OIS, in which case this defaults to the index calendar.

  The `PaymentCalendar` calendar field is currently only supported for `LegType` *Floating* (with an IBOR, BMA or OIS underlying index), *CMS*, *CMSSpread*, *DigitalCMSSpread*, *Equity*, *YY*, *CPI*, *Fixed*, *ZeroCouponFixed*, *DigitalCMS*. For unsupported legs it defaults to the schedule calendar, and if no calendar is set in the `ScheduleData` node (for dates-based schedules the calendar field is optional), the *NullCalendar* is used.

- PaymentConvention: The payment convention of the leg coupons.

  Allowable values: See Table 14.

- PaymentLag [optional]: The payment lag applies to the coupons on Fixed legs, Equity legs, and Floating legs with Ibor and OIS indices (but not to BMA/SIFMA indices), as well as CMS legs, CMSSpread legs, CPI legs and Zero Coupon Fixed legs.
  PaymentLag is also not supported for CapFloor Floating legs that have Ibor coupons with sub periods (HasSubPeriods = *true*), nor for CapFloor Floating legs with averaged ON coupons (IsAveraged = *true*).

  Allowable values: Any valid period, i.e. a non-negative whole number, optionally followed by *D* (days), *W* (weeks), *M* (months), *Y* (years). Defaults to *0D* if left blank or omitted. If a whole number is given and no letter, it is assumed that it is a number of *D* (days).

- NotionalPaymentLag [optional]: The notional payment lag (in days) applied to any notional exchanges.

  Allowable values: Any non-negative integer. Defaults to zero if left blank or omitted.

- DayCounter: The day count convention of the leg coupons. Note that `DayCounter` is mandatory for all leg types except *Equity*.

  Allowable values: See `DayCount Convention` in Table 18. For *Equity* legs, if left blank or omitted, it defaults to *ACT/365*.

- Notionals: This node contains child elements of type `Notional`. If the notional is fixed over the life of the leg only one notional value should be entered. If the notional is amortising or accreting, this is represented by entering multiple

notional values, each represented by a `Notional` child element. The first notional value corresponds to the first coupon, the second notional value corresponds to the second coupon, etc. If the number of coupons exceeds the number of notional values, the notional will be kept flat at the value of last entered notional for the remaining coupons. The number of entered notional values cannot exceed the number of coupons.

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

An example of a `Notionals` element for an amortising leg with four coupons is shown in Listing 176.

Listing 176: Notional list

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

Another allowable specification of the notional schedule is shown in Listing 177.

Listing 177: Notional list with dates

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

The first notional must not have a start date, it will be associated with the schedule's start, The subsequent notionals must either all or none have a start date specified from which date onwards the new notional is applied. This allows specifying notionals only for dates where the notional changes.

An initial exchange, a final exchange and an amortising exchange can be specified using an `Exchanges` child element with `NotionalInitialExchange`, `NotionalFinalExchange` and `NotionalAmortizingExchange` as subelements, see Listing 178. The `Exchanges` element is typically used in cross-currency swaps and inflation swaps, but can also be used in other trade and leg types. Note that for cross-currency swaps, the `NotionalInitialExchange` must be set to the same value on both legs. The `NotionalFinalExchange` must also be set to the same value on both legs, i.e. *true* on both, or *false* on both.

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

Listing 178: Notional list with exchange

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

FX Resets, used for Rebalancing Cross-currency swaps, can be specified using an `FXReset` child element with the following subelements: See Listing 179 for an example.

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

  Allowable values: See Table 15 `Currency`.

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

  Allowable values: Any positive real number.

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

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

Listing 179: Notional list with fx reset

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

- StrictNotionalDates [Optional]: If given and set to true, notional changes specified by startDate will be interpreted as taking place on the exact given date, even if that date falls into a calculation (accrual) period. Otherwise the notional change is applied for the next calculation period. Supported only for fixed and floating legs with IBOR / RFR term rate coupons.
- ScheduleData: This is a trade component sub-node outlined in section 2.3.4 Schedule Data and Dates.
- `PaymentSchedule` [Optional]: This node allows for the specification of an explicit payment schedule, see 2.3.4. Supported in commodity trades, fixed legs and floating legs with underlying OIS and IBOR indices.

- `PaymentDates` [Deprecated]: This node allows for the specification of a list of explicit payment dates. The usage is deprecated, use PaymentSchedule instead.
- FixedLegData: This trade component sub-node is required if `LegType` is set to *Fixed* It is outlined in section 2.3.5.
- FloatingLegData: This trade component sub-node is required if `LegType` is set to *Floating* It is outlined in section 2.3.6 Floating Leg Data and Spreads.
- CashflowLegData: This trade component sub-node is required if `LegType` is set to *Cashflow*. It is outlined in section 2.3.9.
- CMSLegData: This trade component sub-node is required if `LegType` is set to *CMS* (Constant Maturity Swap). It is outlined in section 2.3.10.
- CMBLegData: This trade component sub-node is required if `LegType` is set to *CMB* (Constant Maturity Bond). It is outlined in section 2.3.11.
- DigitalCMSLegData: This trade component sub-node is required if `LegType` is set to *DigitalCMS*. It is outlined in section 2.3.12.
- DurationAdjustedCMSLegData: This trade component sub-node is required if `LegType` is set to *DurationAdjustedCMS*. It is outlined in section 2.3.13.
- CMSSpreadLegData: This trade component sub-node is required if `LegType` is set to *CMSSpread*. It is outlined in section 2.3.14.
- DigitalCMSSpreadLegData: This trade component sub-node is required if `LegType` is set to *DigitalCMSSpread*. It is outlined in section 2.3.15.
- EquityLegData: This trade component sub-node is required if `LegType` is set to *Equity*. It is outlined in section 2.3.16.
- CPILegData: This trade component sub-node is required if `LegType` is set to *CPI*. It is outlined in section 2.3.17.
- YYLegData: This trade component sub-node is required if `LegType` is set to *YY*. It is outlined in section 2.3.18.
- ZeroCouponFixedLegData: This trade component sub-node is required if `LegType` is set to *ZeroCouponFixed*. It is outlined in section 2.3.19.
- FormulaBasedLegData: This trade component sub-node is required if `LegType` is set to *FormulaBased*. It is outlined in section 2.3.35.
- CommodityFloatingLegData: This trade component sub-node is required if `LegType` is set to *CommodityFloating* It is outlined in section 2.3.24.
- CommodityFixedLegData: This trade component sub-node is required if `LegType` is set to *CommodityFixed* It is outlined in section 2.3.21.
- EquityMarginLegData: This trade component sub-node is required if `LegType` is set to *EquityMargin* It is outlined in section 2.3.26.

2.3.4 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 180, Listing 181, and Listing 182 respectively.

Listing 180: Schedule data, rules based

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

Listing 181: Schedule data, date based

```
<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 182: 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 13.
- `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 13.
- `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 17 Calendar.

- `Convention`: Determines the adjustment of the schedule dates with regards to the selected calendar, i.e. the roll convention.

  Allowable values: See Table 14 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 14 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 16 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 29. 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 14 Roll Convention.

  Allowable values: Boolean node, allowing *Y, N, 1, 0, true, false* etc. The full set of allowable values is given in Table 29. 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 13. 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 13. 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 17 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 14 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 29. 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 14 Roll Convention. Defaults to *Preceding* if omitted.

- `IncludeDuplicateDates` [Optional]: If set to *false* the resulting schedule will have unqiue 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 13.

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 17 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 14 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

2.3.5 Fixed Leg Data and Rates

The `FixedLegData` trade component node is used within the `LegData` trade component when the `LegType` element is set to *Fixed*. The `FixedLegData` node only includes the `Rates` sub-node which contains the rates of the fixed leg as child elements of type `Rate`.

An example of a `FixedLegData` node for a fixed leg with constant notional is shown in Listing 183.

Listing 183: Fixed leg data

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

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

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

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

  As in the case of notionals, the rate schedule can be specified with dates as shown in Listing 184.

Listing 184: Fixed leg data with 'dated' rates

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

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

The `FloatingLegData` trade component node is used within the `LegData` trade component when the `LegType` element is set to *Floating*. It is also used directly within the `CapFloor` trade data container. The `FloatingLegData` node includes elements specific to a floating leg.

An example of a `FloatingLegData` node is shown in Listing 185.

Listing 185: Floating leg data

```
<FloatingLegData>
    <Index>USD-LIBOR-3M</Index>
    <IsInArrears>false</IsInArrears>
    <IsAveraged>false</IsAveraged>
    <HasSubPeriods>false</HasSubPeriods>
    <IncludeSpread>false</IncludeSpread>
    <FixingDays>2</FixingDays>
    <Spreads>
        <Spread>0.005</Spread>
    </Spreads>
    <Gearings>
        <Gearing>2.0</Gearing>
    </Gearings>
    <Caps>
        <Cap>0.05</Cap>
    </Caps>
    <Floors>
        <Floor>0.01</Floor>
    </Floors>
    <NakedOption>false</NakedOption>
    <LocalCapFloor>false</LocalCapFloor>
    <HistoricalFixings>
                    <Fixing fixingDate="2016-02-01">0.2</Fixing>
                </HistoricalFixings>
</FloatingLegData>
```

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

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

  Allowable values: An alphanumeric string of the form CCY-INDEX-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. See Table 19. TENOR is not required for Overnight indices, but can be set to *1D*.

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

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

- HasSubPeriods [Optional]: For cases where several Ibor fixings result in a single payment for a period, e.g. if the Ibor tenor is 3M and the schedule tenor is 6M, two fixings are used to compute the amount of the semiannual coupon payments. *true* indicates that an average (IsAveraged = true) or a compounded (IsAveraged=false) value of the fixings is used to determine the payment rate. *false* indicates that the initial index period fixing determines the payment rate for the full tenor, i.e. no further fixings, no averaging and no compounding.

IsAveraged is ignored for Ibor legs when HasSubPeriods is set to *false*. HasSubPeriods does not apply to Overnight indices.

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

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

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

  A Zero Coupon Floating leg with compounding that includes spread can be set up using a rules-based schedule as shown in Listing 186. Note that the `Tenor` in the rules-based schedule is not used when `Rule` is set to *Zero*.

Listing 186: Zero Coupon Floating Leg - Rules-based

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

A Zero Coupon Floating leg with compounding that includes spread can also be set up using a dates-based schedule with two dates (start and end) as shown in Listing 187.

Listing 187: Zero Coupon Floating Leg - Dates-based

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

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

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

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

- LastRecentPeriod [Optional]: Only applies to OIS coupons. If given, the compounding / averaging of ON rates will not be done over the usual reference period derived from the accrual period and the Lookback, FixingDays and IsInArrears parameters, but instead over a period determined by the end date of this usual period and the LastRecentPeriod parameter as [EndDate - LastRecentPeriod, EndDate]. The calendar used to compute EndDate - LastRecentPeriod is the schedule calendar unless a specific LastRecentPeriodCalendar is specified. To represent SOFR 30D, 90D, 180D average indices, the LastRecentPeriodCalendar should be set to NullCalendar, since these averages refer to rolling averages over 30, 90, 180 calendar days.
  Allowable values: any valid period, e.g. 30D, 90D, 180D, 1M, 2M, 6M
- LastRecentPeriodCalendar [Optional]: The calendar used to compute the LastRecentPeriod, see this field for more details. If not given, defaults to the schedule calendar.
  Allowable values: See Table 17 Calendar.
- FixingDays [Optional]: The fixing gap. For Ibor coupons this is the number of business days before the accrual period's *reference* date to observe the index fixing. Here, the accrual period reference date is the accrual start date for an in advanced fixed coupon and the accrual end date for in arrears fixed coupon.
  For overnight coupons this is the number of business days by which the value dates are shifted into the past to get the fixing observation dates. In the context of RFRs the FixingDays parameter is sometimes also called "obervation lag".

  The calendar used for the fixing gap, is the calendar associated with the floating index, as defined in the conventions for the index.

  Allowable values: A non-negative whole number. Defaults to the index's fixing days if blank or omitted. See defaults per index in Table 20.
- Lookback [Optional]: Only applicable to OIS legs. A period by which the value dates schedule of (averaged, compounded) OIS legs is shifted into the past. On top of this the gap defined by the FixingDays is applied to get the final fixing date for an original date in the OIS value dates schedule. In the context of RFRs the Lookback parameter is sometimes also called "shift". With this terminology, first the shift and then the observation lag is applied to get the fixing date for an original value date of an overnight coupon.

  Allowable values: any valid period, e.g. 2D, 3M, 1Y
- RateCutoff [Optional]: Only applicable to OIS legs. The number of fixing dates at the end of the fixing period for which the fixing value is held constant and set to the previous value. Defaults to 0.

  Allowable values: any non-negative whole number

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

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

  For the `<Spreads>` section, the same applies as for notionals and rates - a list of changing spreads can be specified without or with individual start dates as shown in Listing 188.

Listing 188: 'Dated' spreads

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

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

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

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

- Caps [Optional]: This node contains child elements of type `Cap` indicating that the coupon rate is capped at the given rate (after applying gearing and spread, if any). The mode of specification is analogous to spreads, see above. Caps / Floors are supported for Ibor, SIFMA, compounded / averaged OIS coupons, but not for coupons with subperiods.

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

  – IncludeSpread = false, LocalCapFloor = false:

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

  – IncludeSpread = true, LocalCapFloor = false:

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

– IncludeSpread = false, LocalCapFloor = true:

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

– IncludeSpread = true, LocalCapFloor = true:

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

The cases for Averaged coupons are:

– IncludeSpread = false, LocalCapFloor = false:

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

– IncludeSpread = true, LocalCapFloor = false:

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

– IncludeSpread = false, LocalCapFloor = true:

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

– IncludeSpread = true, LocalCapFloor = true:

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

- Floors [Optional]: This node contains child elements of type `Floor` indicating that the coupon rate is floored at the given rate (after applying gearing and spread, if any). The mode of specification is analogous to spreads, see above.
- NakedOption [Optional]: Optional node, if *true* the leg represents only the embedded floor, cap or collar. By convention the embedded floor (or cap) are considered long if the leg is a receiver leg, otherwise short. For a collar the floor is long and the cap is short if the leg is a receiver leg. Notice that this is opposite to the definition of a collar in 2.2.3.

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

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

- `FixingSchedule` [Optional]: This node allows for the specification of an explicit fixing schedule, see 2.3.4. Supported for underlying IBOR / term rate index. A given fixing will become effective as specified by FixingDays relative to the fixing schedule or by an explicit ResetSchedule.
- `ResetSchedule` [Optional]: This node allows for the specification of an explicit reset schedule, see 2.3.4, i.e. the dates on which fixings become effective. Supported for underlying IBOR / term rate index. Can be given together with FixingSchedule or FixingDays. In the latter case, the fixing dates are derived from the reset schedule.
- `HistoricalFixings` [Optional]: This node allows for the specification of an custom trade specific fixings. Supported for underlying OIS / IBOR / term rate index. If a historical fixing for date in the provided list is needed for pricing, the custom fixings will be used instead of an exisiting global index fixings.

2.3.7 Leg Data with Amortisation Structures

Amortisation structures can (optionally) be added to a leg as indicated in the following listing 189, within a block of information enclosed by `<Amortizations>` and `</Amortizations>` tags. Note that `<Amortizations>` structures are not supported for trade type CapFloor.

Listing 189: Amortisation data

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

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

- Type: Amortisation type with allowable values *FixedAmount, RelativeToInitialNotional, RelativeToPreviousNotional, Annuity, LinearToMaturity.*

- Value [optional]: Interpreted depending on `Type`, see below. Required for all types except LinearToMaturity.
- StartDate [optional]: Amortisation starts on first schedule date on or beyond StartDate. If not given, amortisation starts in first schedule period. If more than one AmortizationData block is specified, the StartDate is mandatory for all blocks except the first.
- EndDate [optional]: Amortization is applied for schedule periods with start date before EndDate. If more than one AmortizationData block is specified, the EndDate is mandatory for all blocks except the last.
- Frequency, entered as a period [optional]: Frequency of amortisations. If not given, an amortization is applied in each schedule period, otherwise in each nth period, where n is determined from Frequency. Amortizations are always applied to whole periods though, i.e. not within a period. The frequency is ignored for type Annuity, in which case an amortisation is applied in each period.
- Underflow [optional]: Allow amortisation below zero notional if `true`, otherwise amortisation stops at zero notional. Defaults to false;

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

- FixedAmount: The value is interpreted as a notional amount to be subtracted from the current notional on each amortisation date.
- RelativeToInitialNotional: The value is interpreted as a fraction of the **initial** notional to be subtraced from the current notional on each amortisation date.
- RelativeToPreviousNotional: The value is interpreted as a fraction of the **previous** notional to be subtraced from the previous notional to get the current notional on each amortisation date.
- Annuity: The value is interpreted as annuity amount (redemption plus coupon).
- LinearToMaturity: The value is not relevant, and does not need to be provided.

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

Note:

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

2.3.8 Indexings

This trade component can be used as an optional node within the `LegData` component to scale the notional of the coupons of a leg by one or several index prices. This feature is typically used within equity swaps with notional reset to align the notional of the funding leg with the one from the equity leg for all return periods. See 2.2.25 for the specific usage in equity swaps. Notice that typically it is sufficient to set the `FromAssetLeg` flag to *true* in the `Indexings` node definition, i.e.

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

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

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

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

If specified the notional of the single coupons in the leg is scaled by one or several index prices and a quantity. The indices can be equity or FX indices. Notice that if notional exchanges are enabled on a leg with the `FromAssetLeg` flag set to *true*, the notional exchanges are *not* influenced by the indexing definitions. In general we assume that notional exchanges are not enabled in combination with `FromAssetLeg` *true*, but it is not forbidden technically. Listing 190 shows an example of a Floating leg indexed by both an equity price and a FX rate.

Another use case for Indexings is for non-deliverable IR and Cross Currency Swaps. A non-deliverable IR Swap has Currency set to the deliverable currency on both legs, Notional in the non-deliverable currency on both legs, and Indexings with an FX Index between the deliverable and non-deliverable currency on both legs. See the Swap section for an example non-deliverable IR swap where USD is the payment currency and CLP is the non-deliverable currency.

A non-deliverable Cross Currency Swap has Settlement set to *Cash*, and one leg is a regular leg in the deliverable currency without Indexings. The other leg has Currency set to the deliverable currency, Notional in the non-deliverable currency and Indexings with an FX Index between the deliverable and non-deliverable currency. See the Swap section for an example USD-CLP non-deliverable cross currency swap where CLP is the non-deliverable currency.

Listing 190: Indexings node

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

The Indexings node contains the following elements:

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

  `FromAssetLeg` is supported for the following trade types:

  - `EquitySwap`: Setting `FromAssetLeg` to *true*, aligns the notionals for all return periods on the non-equity funding leg, to the equity leg by deriving equity price, quantity and FX from the equity leg.
    Note that `FromAssetLeg` is only supported if `NotionalReset` is *true* on the equity leg - `FromAssetLeg` is ignored otherwise. Also `FromAssetLeg` is only supported when Quantity is given on the equity leg, not InitialPrice and Notional.

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

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

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

    Allowable values: Any number. Defaults to *1* if left blank or omitted.

  – Index: The relevant index. This is either an equity or FX index. For an FX index, one of the currencies of the index must match the leg currency. It is then ensured that the FX conversion is applied using the correct direction, i.e. if the foreign currency of the index matches the leg currency, the reciprocals of the index fixings are used as a multiplier.

    Allowable values: This is "FX-SOURCE-CCY1-CCY" for FX, see 2.3.29 and 21 for details, or "EQ-NAME" for Equity with "Name" being the general string representation for equity underlyings IdentifierType:Name:Currency:Exchange, see 2.3.29.

  – Dirty [Optional]: Only used for bond indices. Indicates whether to use dirty (*true*) or clean (*false*) prices.

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

  – Relative [Optional]: Only used for bond indices. Indicates whether to use relative (*true*) or absolute prices (*false*). The absolute price is the dirty or clean npv as of the settlement date of the bond in absolute "dollar" terms using the bond details (in particular the notional) from the reference data. The relative price is the absolute price divided by the current notional as of the settlement date.

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

  – ConditionalOnSurvival [Optional]: Only used for bond indices. Indicates whether to forecast bond prices conditional on survival (*true*) or including the default probability from today until the fixing date (*false*).

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

  – InitialFixing [Optional]: If given the index fixing value to apply on the fixing date of the first coupon. If not given the value is read from the relevant fixing history.

    Allowable values: any number

– InitialNotionalFixing [Optional]: If given the index fixing value to apply on the fixing date of the first notional exchange flow. This is used in non-deliverable XCCY swaps. If not given the value is read from the relevant fixing history.

  Allowable values: any number

– ValuationSchedule [Optional]: If given the schedule from which the fixing dates are deduced. If not given, it defaults to the original leg's schedule.

  If the valuation schedule has the same size as the original leg's schedule, it is assumed that the periods correspond one to one, i.e. the ith fixing date is derived from the ith (inArrears = false) or $i+1$th (inArrears = true) date in the valuation schedule using the FixingDays, FixingCalendar and FixingConvention.

  If the valuation schedule has a different size than the original leg's schedule, the relevant valuation date for the ith original leg's coupon is determined as the latest valuation date that is less or equal to accrual start date (inArrears = false) resp. accrual end date (inArrears = true) of that coupon. The fixing date is derived from the relevant valuation date as above, i.e. using the FixingDays, FixingCalendar and FixingConvention.

  Allowable values: a valid schedule definition, see 2.3.4

– FixingDays [Optional]: If given defines the number of fixing days to apply when deriving the fixing dates from the valuation schedule (see above).

  Allowable values: Any non-negative whole number. Defaults to *0* if left blank or omitted.

– FixingCalendar [Optional, defaults to NullCalendar (no holidays): If given defines the fixing calendar to use when deriving the fixing dates from the valuation schedule (see above).

  Allowable values: Allowable values: See Table 17 Calendar. Defaults to the *NullCalendar* (no holidays) if left blank or omitted.

– FixingConvention [Optional]: If given defines the business day convention to use when deriving the fixing dates from the valuation schedule (see above). Defaults to *Preceding* if left blank or omitted.

  Allowable values: Any valid roll convention, e.g. (*F, MF, P, MP, U*). See Table 14.

– IsInArrears [Optional]: If *true*, the fixing dates are derived from the period end dates, otherwise from the period start dates as described for `ValuationSchedule` above.

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

2.3.9 Cashflow Leg Data

A Cashflow leg is used to represent one or more custom cashflows, with specified dates and amounts. Listing 191 shows an example for a leg of type Cashflow.

Listing 191: Cashflow leg data

```
<LegData>
  <Payer>false</Payer>
  <LegType>Cashflow</LegType>
  <Currency>EUR</Currency>
  <PaymentConvention>ModifiedFollowing</PaymentConvention>
  <CashflowData>
    <Cashflow>
      <Amount date="2024-12-15">105000</Amount>
    </Cashflow>
  </CashflowData>
</LegData>
```

The CashflowData block contains the following elements:

- Cashflow: This node contains child elements of type `Amount`, each representing a cashflow. Each child element should include the date of the cashflow using the form:

  ```
  <Amount date="YYYY-MM-DD">[amount]</Amount>
  ```

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

2.3.10 CMS Leg Data

Listing 192 shows an example for a leg of type CMS.

Listing 192: CMS leg data

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

The CMSLegData block contains the following elements:

- Index: The underlying CMS index.

  Allowable values: See 19, a string on the form CCY-CMS-TENOR, where the CMS part stays constant and TENOR is an integer followed by Y.

- Spreads [Optional]: The spreads applied to index fixings. As usual, this can be a single value, a vector of values or a dated vector of values.

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

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

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

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

  Allowable values: A non-negative whole number. Defaults to the fixing days of the Ibor index the swap references if blank or omitted. See defaults per index in

Table 20.

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

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

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

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

- NakedOption [Optional]: If *true* the leg represents only the embedded floor, cap or collar. By convention the embedded floor (or cap) are considered long if the leg is a receiver leg, otherwise short. For a collar the floor is long and the cap is short if the leg is a receiver leg.

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

2.3.11 Constant Maturity Bond Leg Data

In close analogy to the CMS leg one can create a leg that is linked to a *Constant Maturity Bond* yield index. The associated leg type is *CMB*.

Listing 193 shows an example for a leg of type CMB.

Listing 193: CMB leg data

```
<LegData>
  <LegType>CMB</LegType>
  <Payer>false</Payer>
  <Currency>EUR</Currency>
  <Notionals>
    <Notional>10000000</Notional>
  </Notionals>
  <DayCounter>ACT/ACT</DayCounter>
  <PaymentConvention>Following</PaymentConvention>
  <ScheduleData>
    ...
  </ScheduleData>
  <CMBLegData>
    <Index>CMB-US-TBILL-HD-13W</Index>
    <FixingDays>2</FixingDays>
    <Spreads>
      <Spread>0.0010</Spread>
    </Spreads>
    <Gearings>
      <Gearing>2.0</Gearing>
    </Gearings>
  </CMBLegData>
</LegData>
```

The CMBLegData block contains the following elements:

- Index: The underlying CMB index.

  Allowable values: A string of the form CMB-FAMILY-TENOR, where FAMILY might consist of several tokens separated by "-" as e.g. in CMB-US-TBILL-HD or CMB-DE-BUND, and TENOR is a valid period.

- Spreads [Optional]: The spreads applied to index fixings. As usual, this can be a single value, a vector of values or a dated vector of values.

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

- FixingDays: This is the fixing gap, i.e. the number of days before the period end date an index fixing is taken. Defaults to the index's fixing gap.

  Allowable values: A non-negative whole number. Defaults to the fixing days of the Ibor index the swap references if blank or omitted. See defaults per index in Table 20.

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

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

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

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

Note:

- For each CMB index name one needs to maintain bond reference data with ID equal to the index name. This reference data is used to project the CMB index fixings as follows: For a future index period (from future date to future date + index tenor), a forward starting bond is constructed using the schedule frequency defined in the reference data and with constant maturity (future date + tenor). The forward bond is priced using the reference yield curve and credit curve defined in the reference data. And the bond price is then converted into a bond yield using the bond yield conventions (compounding, frequncy, price type) maintained for that same ID. If the conventions are not set up, then default values are used (compounded, annual, clean).

- For periods with start dates in the past, historical index fixings need to be provided, as for interest rate indices.

- No convexity adjustment is applied here yet, in contrast to CMS index projections.

- The CMB leg does not support Caps or Floors yet, in contrast to the CMS leg.

2.3.12 Digital CMS Leg Data

Listing 194 shows an example for a leg of type *DigitalCMS*.

Listing 194: Digital CMS leg data

```
<LegData>
  <LegType>DigitalCMS</LegType>
  <Payer>false</Payer>
  <Currency>GBP</Currency>
  <Notionals>
    <Notional>10000000</Notional>
  </Notionals>
  <DayCounter>ACT/ACT</DayCounter>
  <PaymentConvention>Following</PaymentConvention>
  <ScheduleData>
    ...
  </ScheduleData>
  <DigitalCMSLegData>
    <CMSLegData>
      <Index>EUR-CMS-10Y</Index>
      <FixingDays>2</FixingDays>
      <Gearings>
        <Gearing>3.0</Gearing>
      </Gearings>
      <Spreads>
        <Spread>0.0010</Spread>
      </Spreads>
      <NakedOption>false</NakedOption>
    </CMSLegData>
    <CallPosition>Long</CallPosition>
    <IsCallATMIncluded>false</IsCallATMIncluded>
    <CallStrikes>
      <Strike>0.003</Strike>
    </CallStrikes>
    <CallPayoffs>
      <Payoff>0.003</Payoff>
    </CallPayoffs>
    <PutPosition>Short</PutPosition>
    <IsPutATMIncluded>false</IsPutATMIncluded>
    <PutStrikes>
      <Strike>0.05</Strike>
    </PutStrikes>
    <PutPayoffs>
      <Payoff>0.05</Payoff>
    </PutPayoffs>
  </DigitalCMSLegData>
</LegData>
```

The `DigitalCMSLegData` block contains the following elements:

- CMSLegData: a `CMSLegData` block describing the underlying Digital CMS leg (see 2.3.10). Caps and floors in the underlying CMS leg are not supported for Digital CMS Options. The `NakedOption` flag in the `CMSLegData` block is supported and can be used to separate the digital option payoff from the underlying CMS coupon.

- CallPosition: Specifies whether the call option position is long or short.
- IsCallATMIncluded: inclusion flag on the call payoff if the call option ends at-the-money
- CallStrikes: strike rate for the call option
- CallPayoffs: digital call option payoff rate. If included the option is cash-or-nothing, if excluded the option is asset-or-nothing
- PutPosition: Specifies whether the put option position is long or short.
- IsPutATMIncluded: inclusion flag on the put payoff if the put option ends at-the-money
- PutStrikes: strike rate for the put option
- PutPayoffs: digital put option payoff rate. If included the option is cash-or-nothing, if excluded the option is asset-or-nothing

2.3.13 Duration Adjusted CMS Leg Data

Listing 195 shows an example for a leg of type DurationAdjustedCMS.

Listing 195: Duration Adjusted CMS leg data

```
<LegData>
  <LegType>DurationAdjustedCMS</LegType>
  <Payer>false</Payer>
  <Currency>EUR</Currency>
  <Notionals>
    <Notional>21000000</Notional>
  </Notionals>
  <DayCounter>Simple</DayCounter>
  <PaymentConvention>Following</PaymentConvention>
  <PaymentLag>0</PaymentLag>
  <PaymentCalendar>TARGET</PaymentCalendar>
  <DurationAdjustedCMSLegData>
    <Index>EUR-CMS-20Y</Index>
    <Duration>10</Duration>
    <Spreads>
      <Spread>0</Spread>
    </Spreads>
    <Gearings>
      <Gearing>1</Gearing>
    </Gearings>
    <IsInArrears>false</IsInArrears>
    <FixingDays>2</FixingDays>
    <Caps>
      <Cap>0.015</Cap>
    </Caps>
    <NakedOption>true</NakedOption>
  </DurationAdjustedCMSLegData>
  <ScheduleData>
    <Rules>
      <StartDate>2019-12-31</StartDate>
      <EndDate>2023-12-31</EndDate>
      <Tenor>3M</Tenor>
      <Calendar>EUR</Calendar>
      <Convention>ModifiedFollowing</Convention>
      <TermConvention>ModifiedFollowing</TermConvention>
      <Rule>Backward</Rule>
    </Rules>
  </ScheduleData>
</LegData>
```

The DurationAdjustedCMSLegData is identical to the CMSLegData block (see 2.3.10). In addition to this it contains a field defining the duration adjustment:

- Duration [Optional]: A non-negative whole number n that defines the duration adjustment for the coupons. If γ is the underlying CMS index fixing for a coupon the duration adjustment δ is defined as

$$\delta = \sum_{i=1}^{n} \frac{1}{(1+\gamma)^i}$$

  If n is zero or the duration is not given, δ is defined as 1, i.e. no adjustment is applied in this case and the coupon will be an ordinary CMS coupon. The coupon amount A for a duration adjusted coupon with a spread s, a gearing g, a

cap c, a floor f and with nominal N and accrual fraction τ is given by:

$$A = \delta \cdot N \cdot \tau \cdot \max(\min(g \cdot \gamma + s, c), f)$$

Allowable values: A non-negative whole number.

2.3.14 CMS Spread Leg Data

A CMS Spread leg consists of coupons that are linked to differences (spreads) of CMS index fixings with different maturities. When these are capped and/or floored, then the leg (assuming NakedOption *true*) contains a sequence of CMS Spread Options with payoff

$$N \cdot [(CMS_m - CMS_n - K)]^+ = N \cdot \max(0, CMS_m - CMS_n - K)$$

where N is the notional amount, $CMS_{n/m}$ is the n/m year CMS rate, and K is the strike.

Listing 196 shows an example for a leg of type CMSSpread.

Listing 196: CMS Spread leg data

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

The elements of the CMSSpreadLegData block are identical to those of the CMSLegData (see 2.3.10), except for the index which is defined by two CMS indices as the difference between `Index1` and `Index2`.

The payout for each coupon is thus:

$N \cdot (gearing \cdot (Index1 - Index2) + Spread) \cdot daycountfraction$

Adding a cap, and assuming no spread, a gearing of 1, a daycount fraction of 1, and a notional of 1, the payout becomes:

$min(Cap; Index1 - Index2)$

If there is a floor instead of a cap, the payout is:

$max(Floor; Index1 - Index2)$

Note that a CMS Spread Option can be created by setting NakedOption to *true*. With this setting, the payout for the CMS Spread leg with a cap becomes an option with the cap rate as strike:

$max(0; (Index1 - Index2) - Cap)$

And the payout for a CMS Spread leg with a floor, and with NakedOption set to *true* is :

$max(0; Floor - (Index1 - Index2))$

2.3.15 Digital CMS Spread Leg Data

Listing 197 shows an example for a leg of type *DigitalCMSSpread*.

Listing 197: Digital CMS Spread leg data

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

The `DigitalCMSSpreadLegData` block contains the following elements:

- CMSSpreadLegData: a `CMSSpreadLegData` block describing the underlying Digital CMS Spread leg (see 2.3.14). Caps and floors in the underlying CMS Spread leg are not supported for Digital CMS Spread Options. The `NakedOption` flag in the `CMSSpreadLegData` block is supported and can be used to separate the digital option payoff from the underlying CMS Spread coupon.
- CallPosition: Specifies whether the call option position is long or short.
- IsCallATMIncluded: inclusion flag on the call payoff if the call option ends at-the-money

- CallStrikes: strike rate for the call option
- CallPayoffs: digital call option payoff rate. If included the option is cash-or-nothing, if excluded the option is asset-or-nothing
- PutPosition: Specifies whether the put option position is long or short.
- IsPutATMIncluded: inclusion flag on the put payoff if the put option ends at-the-money
- PutStrikes: strike rate for the put option
- PutPayoffs: digital put option payoff rate. If included the option is cash-or-nothing, if excluded the option is asset-or-nothing

2.3.16 Equity Leg Data

Listing 198 shows an example of a leg of type Equity. Note that a resetting Equity Leg (NotionalReset set to *true*) must have either:
a) a Quantity, or
b) an InitialPrice and a Notional in the leg

The EquityLegData block contains the following elements:

- Quantity[Optional with one exception]: The number of shares. Either a Notional or the Quantity must be given for the leg, but not both at the same time.

  Quantity is optional with the exception that when FXTerms is used and NotionalReset is set to *true*, and the InitialPriceCurrency differs from the leg currency, Quantity must be given, and Notional cannot be used.

  Allowable values: Any positive real number

- ReturnType: *Price* indicates that the coupons on the equity leg are determined by the price movement of the underlying equity, i.e.:
  $Notional \cdot \frac{FinalPrice - InitialPrice}{InitialPrice}$,
  Total indicates that coupons are determined by the total return of the underlying equity including dividends, i.e.:
  $Notional \cdot \frac{(FinalPrice + dividends * DividendFactor) - InitialPrice}{InitialPrice}$,
  Dividend indicates that the coupons are determined by the dividened paid on the underlying equity.

  Allowable values: *Price, PRICE*, *Total, TOTAL*, *Dividend, DIVIDEND*

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

  Allowable values: See `Name` for equity trades in Table 24.

- Underlying: This node may be used as an alternative to the `Name` node to specify the underlying equity. This in turn defines the equity curve used for pricing. The `Underlying` node is described in further detail in Section 2.3.29.

- InitialPrice [Optional]: Initial Price of the equity, if not present, the first valuation date is used to determine the initial price. If InitialPrice is zero then each coupon's price is just the discounted fixing from the coupon's FixingEndDate. For any divisions we assume the value is one, i.e. when

NotionalReset = true we have instead Quantity = Notional. The Initial price can be either given in the currency of the equity or in the leg currency, see InitialPriceCurrency.

Allowable values: Any positive real number. If omitted or left blank it defaults to the equity price of the fixing at the valuation date associated with the start date. Note that when this valuation date is in the future the forward equity price is used.

- InitialPriceCurrency [Optional]: If an initial price is given, it can be either given in the original equity ccy or the leg currency (if these are different). This field determines in which currency the initial price is given. If omitted, it is assumed that the initial price is given in equity currency.

  Allowable values: A valid currency code, See Fiat Currencies and Minor Currencies in Table 15.

- NotionalReset [Optional]: Defaults to *true*. Notional resets only affect the equity leg. If NotionalReset is set to *true* the quantity or number of shares of the underlying equity is fixed for all the coupons on the equity leg and the Notional for a period is computed as

  Notional = Quantity x (share price at valuation date for period) x (FX conversion rate at valuation date for period)

  Notice that either a) the Quantity or b) a Notional and an explicit InitialPrice must be given in the leg data for a resettable leg. In the latter case the Quantity is computed as

  Quantity = Notional / InitialPrice

  No FX conversion is allowed if the Quantity has to be derived from the Notional and the InitialPrice.

  If NotionalReset is set to *false* the quantity of the underlying equity varies per period, as per:

  Quantity = Notional / (Equity Price at valuation date for the period)

  For the first period, the InitialPrice is the Equity Price at valuation date. Here, the Notional is taken to be the Notional specified in the leg or - if the Quantity is given - to be

  Notional = Quantity x InitialPrice

  where again the InitialPrice must be explicitly given in the leg data and no FX conversion is allowed in this case.

  Allowable values: *true* or *false*

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

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

- ValuationSchedule [Optional]: Schedule of dates for equity valuation. If used, fixing dates for equity valuation will come from the `ValuationSchedule` instead of the `ScheduleData`.

  Allowable values: A node of the same form as `ScheduleData`, (see 2.3.4). Note that the number of dates (and periods) in the `ValuationSchedule` must be the same as in the `ScheduleData`. If omitted, equity valuation dates follow the schedule of the equity leg adjusted for FixingDays.

- FixingDays [Optional]: The number of days before payment date for equity valuation. *N.B.* Only used when no valuation schedule present. The calendar used when applying the fixing days is the equity curve calendar (defined in the equity reference data for the underlying equity) combined with the FxIndex calendar (if applicable).

  Allowable values: Any non-negative integer. Defaults to *0* if left blank or omitted.

- FXTerms [Mandatory when leg and equity currencies differ]: For the case when the currency the underlying equity is quoted in, is different from the leg currency. The `FXTerm` node contains the following elements:

  – EquityCurrency [Mandatory within `FXTerms`]: Currency underlying equity is quoted in. Required if FXTerms is present.

    Allowable values: See Fiat Currencies and Minor Currencies in Table 15.

  – FXIndex [Mandatory within `FXTerms`]: Name of the index for FX fixings for the leg vs equity currency pair, e.g. FX-TR20H-EUR-USD for Thomson Reuters 20:00 EURUSD FX fixing. Required if FXTerms present.

    Allowable values: See Table 21

Listing 198: Equity leg data

```
<LegData>
  <LegType>Equity</LegType>
  <Payer>false</Payer>
  <Currency>EUR</Currency>
  <DayCounter>ACT/ACT</DayCounter>
  <PaymentConvention>Following</PaymentConvention>
  <ScheduleData>
    <Rules>
      <StartDate>2026-03-01</StartDate>
      <EndDate>2028-03-01</EndDate>
      <Tenor>3M</Tenor>
      <Calendar>TARGET</Calendar>
      <Convention>ModifiedFollowing</Convention>
      <Rule>Forward</Rule>
    </Rules>
  </ScheduleData>
  <EquityLegData>
    <Quantity>1000.0</Quantity>
    <ReturnType>Price</ReturnType>
    <Underlying>
      <Type>Equity</Type>
      <Name>.SPX</Name>
      <IdentifierType>RIC</IdentifierType>
    </Underlying>
    <InitialPrice>100</InitialPrice>
    <NotionalReset>true</NotionalReset>
    <DividendFactor>1</DividendFactor>
    <ValuationSchedule>
      <Dates>
        <Calendar>USD</Calendar>
        <Convention>ModifiedFollowing</Convention>
        <Dates>
          <Date>2026-03-01</Date>
          <Date>2026-06-01</Date>
          <Date>2026-09-01</Date>
          <Date>2026-12-01</Date>
          <Date>2027-03-01</Date>
          <Date>2027-06-01</Date>
          <Date>2027-09-01</Date>
          <Date>2027-12-01</Date>
          <Date>2028-03-01</Date>
        </Dates>
      </Dates>
    </ValuationSchedule>
    <FixingDays>0</FixingDays>
    <FXTerms>
      <EquityCurrency>USD</EquityCurrency>
      <FXIndex>FX-TR20H-EUR-USD</FXIndex>
    <FXTerms>
  </EquityLegData>
</LegData>
```

2.3.17 CPI Leg Data

A CPI leg contains a series of CPI-linked coupon payments $N\,r\,(I(t)/I_0)\,\delta$ and, if `NotionalFinalExchange` is set to *true*, a final inflation-linked redemption $(I(t)/I_0)\,N$. Each coupon and the final redemption can be subtracting the (un-inflated) notional N, i.e. $(I(t)/I_0 - 1)\,N$, see below.

Note that CPI legs with just a final redemption and no coupons, can be set up with a dates-based Schedule containing just a single date - representing the date of the final redemption flow. In this case `NotionalFinalExchange` must be set to *true*, otherwise the whole leg is empty, and the Rate is not used and can be set to any value.

Listing 199 shows an example for a leg of type CPI with annual coupons, and 200 shows an example for a leg of type CPI with just the final redemption.

The `CPILegData` block contains the following elements:

- Index: The underlying zero inflation index.

  Allowable values: See `Inflation CPI Index` in Table 22.

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

  Note that a CPI leg coupon payment at time t is:

  $$N\,r\,\frac{I(t)}{I_0}\,\delta$$

  where:

  - N: notional
  - r: the contractual fixed real rate
  - $I(t)$: the relevant CPI fixing for time t
  - I_0: the BaseCPI
  - δ: the day count fraction for the accrual period up to time t

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

- BaseCPI [Optional]: The base CPI value I_0 used to determine the lifting factor for the fixed coupons. If omitted it will take the observed CPI fixing on startDate - observationLag.

  Allowable values: Any positive real number.

- StartDate [Optional]: The start date needs to be provided in case the schedule comprises only a single date. If the schedule has at least two dates and a start date is given at the same time, the first schedule date is taken as the start date and the supplied `StartDate` is ignored.

  Allowable values: See `Date` in Table 13.

- ObservationLag [Optional]: The observation lag to be applied. It's the amount of time from the fixing at the start or end of the period, moving backward in time,

to the inflation index observation date (the inflation fixing). Fallback to the index observation lag as specified in the inflation swap conventions of the underlying index, if not specified.

Allowable values: An integer followed by *D*, *W*, *M* or *Y*. Interpolation lags are typically expressed in a positive number of *M*, months. Note that negative values are allowed, but mean that the inflation is observed forward in time from the period start/end date, which is unusual.

- Interpolation [Optional]: The type of interpolation that is applied to inflation fixings. *Linear* interpolation means that the inflation fixing for a given date is interpolated linearly between the surrounding - usually monthly - actual fixings, whereas with *Flat* interpoltion the inflation fixings are constant for each day at the value of the previous/latest actual fixing (flat forward interpolation). Fallback to the Interpolation as specified in the inflation swap conventions of the underlying index, if not specified.

  Allowable values: *Linear, Flat*

- SubtractInflationNotional [Optional]: A flag indicating whether the non-inflation adjusted notional amount should be subtracted from the final inflation-adjusted notional exchange at maturity. Note that the final coupon payment is not affected by this flag.
  Final notional payment if *true*: $N\,(I(T)/I_0 - 1)$.
  Final notional payment if *false*: $N\,I(T)/I_0$

  Allowable values: Boolean node, allowing *Y, N, 1, 0, true, false* etc. The full set of allowable values is given in Table 29.
  Defaults to *false* if left blank or omitted.

- SubtractInflationNotionalAllCoupons [Optional]: A flag indicating whether the non-inflation adjusted notional amount should be subtracted from all coupons. Note that the final redemption payment is not affected by this flag.
  Coupon payment if *true*: $N\,(I(T)/I_0 - 1)$.
  Coupon payment if *false*: $N\,I(T)/I_0$

  Allowable values: Boolean node, allowing *Y, N, 1, 0, true, false* etc. The full set of allowable values is given in Table 29.
  Defaults to *false* if left blank or omitted.

- Caps [Optional]: This node contains child elements of type `Cap` indicating that the inflation indexed payment is capped; the cap is applied to the inflation index and expressed as an inflation rate, see CPI Cap/Floor in the Product Description. If the cap is constant over the life of the cpi leg, only one cap value should be entered. If two or more coupons have different caps, multiple cap values are required, each represented by a `Cap` child element. The first cap value corresponds to the first coupon, the second cap value corresponds to the second coupon, etc. If the number of coupons exceeds the number of cap values, the cap will be kept at at the value of last entered spread for the remaining coupons. The number of entered cap values cannot exceed the number of coupons. Notice that the caps defined under this node only apply to the cpi coupons, but not a final notional flow (if present). A cap for the final notional flow can be defined under

the FinalFlowCap node.

Allowable values: Each child element can take any real number. The cap is expressed in decimal form, e.g. 0.03 is a cap of 3%.

- Floors [Optional]: This node contains child elements of type `Floor` indicating that the inflation indexed payment is floored; the floor is applied to the inflation index and expressed as an inflation rate. The mode of specification is analogous to caps, see above. Notice that the floors defined under this node only apply to the cpi coupons, but not a final notional flow (if present). A floor for the final notional flow can be defined under the FinalFlowFloor node.

  Allowable values: Each child element can take any real number. The floor is expressed in decimal form, e.g. 0.01 is a cap of 1%.

- FinalFlowCap [Optional]: The cap to be applied to the final notional flow of the cpi leg. If not given, no cap is applied.

  Note that final and non-final inflation cap/floor strikes are quoted as a number K and converted to a price via:

  $(1 + K)^t$

  where

  K = the cap/floor rate

  t = time to expiry.

  So inflation caps/floors are caps/floors on the inflation rate and not the inflation index ratio. For example, to cap the final flow at the initial notional it should be K=0, i.e. FinalFlowCap should be 0.

  Allowable values: A real number. The FinalFlowCap is expressed in decimal form, e.g. 0.01 is a cap on the final flow at 1% of the inflation rate over the life of the trade.

- FinalFlowFloor [Optional]: The floor to be applied to the final notional flow of the cpi leg. If not given, no floor is applied.

  Allowable values: A real number. The FinalFlowFloor is expressed in decimal form, e.g. 0.01 is a floor on the final flow at 1% of the inflation rate over the life of the trade.

- NakedOption [Optional]: Optional node, if *true* the leg represents only the embedded floor, cap or collar. By convention these embedded options are considered long if the leg is a receiver leg, otherwise short.

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

Whether the leg cotains a final redemption flow at all or not depends on the notional exchange setting, see section 2.3.3 and listing 178.

Listing 199: CPI leg data with capped annual coupons

```
<LegData>
  <LegType>CPI</LegType>
  <Payer>false</Payer>
  <Currency>GBP</Currency>
  <Notionals>
    <Notional>10000000</Notional>
    <Exchanges>
      <NotionalInitialExchange>false</NotionalInitialExchange>
      <NotionalFinalExchange>true</NotionalFinalExchange>
    </Exchanges>
  </Notionals>
  <DayCounter>ACT/ACT</DayCounter>
  <PaymentConvention>Following</PaymentConvention>
  <ScheduleData>
    <Rules>
      <StartDate>2016-07-18</StartDate>
      <EndDate>2021-07-18</EndDate>
      <Tenor>1Y</Tenor>
      <Calendar>UK</Calendar>
      <Convention>ModifiedFollowing</Convention>
      <TermConvention>ModifiedFollowing</TermConvention>
      <Rule>Forward</Rule>
      <EndOfMonth/>
      <FirstDate/>
      <LastDate/>
    </Rules>
  </ScheduleData>
  <CPILegData>
    <Index>UKRPI</Index>
    <Rates>
      <Rate>0.02</Rate>
    </Rates>
    <BaseCPI>210</BaseCPI>
    <StartDate>2016-07-18</StartDate>
    <ObservationLag>2M</ObservationLag>
    <Interpolation>Linear</Interpolation>
    <Caps>
       <Cap>0.03</Cap>
    </Caps>
    <Floors>
      <Floor>0.0</Floor>
    <Floors>
    <FinalFlowCap>0.03</FinalFlowCap>
    <FinalFlowFloor>0.0</FinalFlowFloor>
    <NakedOption>false</NakedOption>
    <SubtractInflationNotionalAllCoupons>false</SubtractInflationNotionalAllCoupons>
  </CPILegData>
</LegData>
```

Listing 200: CPI leg data with just the final redemption

```
<LegData>
  <Payer>false</Payer>
  <LegType>CPI</LegType>
  <Currency>GBP</Currency>
  <PaymentConvention>ModifiedFollowing</PaymentConvention>
  <DayCounter>ActActISDA</DayCounter>
  <Notionals>
    <Notional>25000000.0</Notional>
    <Exchanges>
      <NotionalInitialExchange>false</NotionalInitialExchange>
      <NotionalFinalExchange>true</NotionalFinalExchange>
    </Exchanges>
  </Notionals>
  <ScheduleData>
    <Dates>
      <Calendar>GBP</Calendar>
      <Dates>
        <Date>2020-08-17</Date>
      </Dates>
    </Dates>
  </ScheduleData>
  <CPILegData>
    <Index>UKRPI</Index>
    <Rates>
      <Rate>1.0</Rate>
    </Rates>
    <BaseCPI>280.64</BaseCPI>
    <StartDate>2018-08-19</StartDate>
    <ObservationLag>2M</ObservationLag>
    <Interpolation>Linear</Interpolation>
    <SubtractInflationNotional>true</SubtractInflationNotional>
    <SubtractInflationNotionalAllCoupons>false</SubtractInflationNotionalAllCoupons>
  </CPILegData>
</LegData>
```

2.3.18 YY Leg Data

Listing 201 shows an example for a leg of type YY. The YYLegData block contains the following elements:

- Index: The underlying zero inflation index.

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

- FixingDays: The number of fixing days.

  Allowable values: An integer followed by D,

- ObservationLag [Optional]: The observation lag to be applied. Fallback to the index observation lag as specified in the inflation swap conventions of the underlying index, if not specified.

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

- Spreads [Optional]: The spreads applied to index fixings. As usual, this can be a single value, a vector of values or a dated vector of values.
- Gearings [Optional]: This node contains child elements of type `Gearing` indicating that the coupon rate is multiplied by the given factors. The mode of specification is analogous to spreads, see above.
- Caps [Optional]: This node contains child elements of type `Cap` indicating that the coupon rate is capped at the given rate (after applying gearing and spread, if any).
- Floors [Optional]: This node contains child elements of type `Floor` indicating that the coupon rate is floored at the given rate (after applying gearing and spread, if any).
- NakedOption [Optional]: Optional node (defaults to N), if Y the leg represents only the embedded floor, cap or collar. By convention these embedded options are considered long if the leg is a receiver leg, otherwise short.
- AddInflationNotional [Optional]: If true, the payoff will include the notional of the coupon $N\,\tau\,\frac{I_t}{I_{t-1Y}}$.
- IrregularYoY [Optional]: If true, instead of using a YoY inflation rate the coupon is based on the inflation rate during the actual coupon period, e.g. for a 6M coupon the inflation rate will be computed as $\frac{I_t}{I_{t-6m}} - 1$.

Listing 201: YY leg data

```
<LegData>
  <LegType>YY</LegType>
  <Payer>false</Payer>
  <Currency>EUR</Currency>
  <Notionals>
    <Notional>10000000</Notional>
  </Notionals>
  <DayCounter>ACT/ACT</DayCounter>
  <PaymentConvention>Following</PaymentConvention>
  <ScheduleData>
    <Rules>
      <StartDate>2016-07-18</StartDate>
      <EndDate>2021-07-18</EndDate>
      <Tenor>1Y</Tenor>
      <Calendar>UK</Calendar>
      <Convention>ModifiedFollowing</Convention>
      <TermConvention>ModifiedFollowing</TermConvention>
      <Rule>Forward</Rule>
      <EndOfMonth/>
      <FirstDate/>
      <LastDate/>
    </Rules>
  </ScheduleData>
  <YYLegData>
    <Index>EUHICPXT</Index>
    <FixingDays>2</FixingDays>
    <ObservationLag>2M</ObservationLag>
    <Interpolated>true</Interpolated>
    <Spreads>
      <Spread>0.0010</Spread>
    </Spreads>
    <Gearings>
      <Gearing>2.0</Gearing>
    </Gearings>
    <Caps>
      <Cap>0.05</Cap>
    </Caps>
    <Floors>
      <Floor>0.01</Floor>
    </Floors>
    <NakedOption>N</NakedOption>
    <AddInflationNotional>false</AddInflationNotional>
                                  <IrregularYoY>false</IrregularYoY>
  </YYLegData>
</LegData>
```

2.3.19 ZeroCouponFixed Leg Data

A Zero Coupon Fixed leg contains a series of Zero Coupon payments, i.e $(1+r)^t N$ for a compounded coupon. The uninflated notional N can be subtracted from the payment , i.e $((1+r)^t - 1) N$, see `SubtractNotional` below.

Listing 202 shows an example for a leg of type Zero Coupon Fixed.

To create a leg with only one payment, the schedule must only contain the start and

end date. Note that this can be achieved by setting the *Tenor* to *0D* or *1T* in a rules based Schedule.

Note that the DayCounter in a Zero Coupon Fixed leg is used to compute t in $(1+r)^t$ so that the series of zero coupon payments are calculated as $(1+r)^t N$. Also note that the "1/1" DayCounter is treated as "Year" DayCounter on Zero Coupon Fixed legs, (otherwise the t would always be 1).

For leg types other than Zero Coupon Fixed, the DayCounter is used to compute the "Accrual" (i.e. the accrual time period of a coupon) in a coupon payment calculated as: $N * Accrual * r$. However, the "Accrual" in the coupon formula is defaulted to 1 for the Zero Coupon Fixed leg type.

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

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

- Compounding [Optional]: The method of compounding applied to the rate.

  Allowable values: *Simple* i.e. $(1+r*t)$, or *Compounded* i.e. $(1+r)^t$. Defaults to *Compounded* if left blank or omitted.

- SubtractNotional [Optional]: Decides whether the notional is subtracted from the compounding factor, i.e. $(1+r*t)-1$ respectively $(1+r)^t-1$, or not, i.e. $(1+r*t)$ respectively $(1+r)^t$

  Note that if NotionalFinalExchange is set to *true* an additional final uninflated notional flow N is added. So if NotionalFinalExchange is set to *true*, and SubtractNotional is set to *false*, there will be two final notional flows. It is recommended to omit NotionalFinalExchange causing it to default to *false*, and solely use SubtractNotional to determine the final notional flow.

  Allowable values: *true*, *Y* or *false*, *N*, defaults to *true* if left blank or omitted.

Listing 202: ZeroCouponFixed leg data

```
<LegData>
  <LegType>ZeroCouponFixed</LegType>
  <Payer>false</Payer>
  <Currency>EUR</Currency>
  <Notionals>
    <Notional>10000000</Notional>
  </Notionals>
  <DayCounter>Year</DayCounter>
  <PaymentConvention>Following</PaymentConvention>
  <ScheduleData>
    <Rules>
      <StartDate>2016-07-18</StartDate>
      <EndDate>2021-07-18</EndDate>
      <Tenor>0D</Tenor>
      <Calendar>UK</Calendar>
      <Convention>ModifiedFollowing</Convention>
      <TermConvention>ModifiedFollowing</TermConvention>
      <Rule>Forward</Rule>
      <EndOfMonth/>
      <FirstDate/>
      <LastDate/>
    </Rules>
  </ScheduleData>
  <ZeroCouponFixedLegData>
      <Rates>
          <Rate>0.02</Rate>
      </Rates>
      <Compounding>Simple</Compounding>
      <SubtractNotional>false</SubtractNotional>
  </ZeroCouponFixedLegData>
</LegData>
```

2.3.20 Commodity Fixed Leg

A commodity fixed leg is specified in a `LegData` node with `LegType` set to `CommodityFixed`. It is used to define a sequence of cashflows that are linked to a fixed price in a commodity derivative contract. Each cashflow has an associated *Calculation Period*. The outline of a commodity fixed leg is given in listing 203. It has the usual `LegData` elements described in section 2.3.3 and a `CommodityFixedLegData` node that is described in section 2.3.21 below. The section 2.3.23 describes some aspects of the `ScheduleData` node in the context of commodity derivatives.

2.3.21 Commodity Fixed Leg Data

The `CommodityFixedLegData` node outline is shown in listing 204. The meaning and allowable values for each node are as follows:

- `Quantities` [Optional]: this node is used to specify a constant quantity or a quantity that varies over the calculation periods. The usage of this node is analagous to the usage of the `Notionals` node as outlined in section 2.3.3. For convenience, this node can be omitted if the quantities are identical to those on a commodity floating leg, outlined in Section 2.3.22, on the same trade. In this

case, the quantities from the floating leg are used. If there is only a single commodity floating leg, as is the case in a standard swap, the quantities are taken from that leg. If there are multiple commodity floating legs on the trade, a specific commodity floating leg can be picked using the `Tag` node specified below. In other words, a `Tag` can be specified on the fixed leg and the same `Tag` specified on the floating leg from which the quantities should be taken.

- `Prices`: this node is used to specify a constant price or a price that varies over the calculation periods. The usage of this node is analagous to the usage of the `Notionals` node as outlined in section 2.3.3.
- `CommodityPayRelativeTo` [Optional]: the allowable values for this node are `CalculationPeriodStartDate`, `CalculationPeriodEndDate`, `TerminationDate`, `FutureExpiryDate`. They specify whether payment is relative to the calculation period start date, calculation period end date, leg maturity date or the future expiry date (of the corresponding cashflow on the floating leg with the same Tag as the fixed leg) respectively. The default is `CalculationPeriodEndDate`. The payment date is then further adjusted by the payment conventions outlined in section 2.3.3 i.e. `PaymentConvention` and `PaymentLag`. If explicit payment dates are given via the `PaymentDates` node described in section 2.3.3, then those explicit payment dates are used instead and adjusted by the `PaymentCalendar` and `PaymentConvention`.
- `Tag` [Optional]: The use of this node is explained in the `Quantities` resp. `CommodityPayRelativeTo` piece above.

2.3.22 Commodity Floating Leg

A commodity floating leg is specified in a `LegData` node with `LegType` set to `CommodityFloating`. It is used to define a sequence of cashflows that are linked to the price of a given commodity. Each cashflow has an associated *Calculation Period*. The price that is being referenced may be a commodity spot price or a commodity future contract settlement price. The cashflow may depend on the price observed on a single *Pricing Date* in the *Calculation Period* or it may depend on the arithmetic average of the prices over some or all of the business days in the *Calculation Period*.

The outline of a commodity floating leg is given in listing 205. It has the usual `LegData` elements described in section 2.3.3 and a `CommodityFloatingLegData` node that is described in section 2.3.24 below. Before describing the `CommodityFloatingLegData` node, we devote section 2.3.23 to the `ScheduleData` node in the context of commodity derivatives.

2.3.23 Commodity Schedules

The *Calculation Period* in a commodity derivative contract is in general specified as a period from and including a given *Start Date* to and including a given *End Date*. A commodity trade leg consists of a sequence of these *Calculation Period*s. It is important to set up the `ScheduleData` in the trade XML such that these periods are correctly represented in the ORE instrument. The `ScheduleData` allows for the creation of a list of dates that define the boundaries of the periods from the trade *Effective Date* to the trade *Termination Date*. When the `ScheduleData` is used on a

commodity leg in the ORE trade XML, the `StartDate` is included in the first period and the `EndDate` is included in the final period. Each intervening date generated by the `ScheduleData` is understood to be the included end date of a period with the subsequent period beginning on the day after the intervening date. The following two examples illustrate the set up of the `ScheduleData`.

A common commodity derivative schedule is one that has monthly periods running from and including the first calendar day in the month to and including the last calendar day in the month. For example, the contract periods may be specified as shown in table 8. The corresponding `ScheduleData` node that should be used to represent this in ORE XML is shown in listing 206. Note that `Convention` and `TermConvention` are set to `Unadjusted` and `EndOfMonth` is set to `true` to place all dates at the end of the month when generating the dates `Backward` from 30 Apr 2020. In general, these values should be used when generating monthly periods for commodity derivatives.

Start Date	End Date
2020-01-01	2020-01-31
2020-02-01	2020-02-29
2020-03-01	2020-03-31
2020-04-01	2020-04-30

Table 8: Commodity derivative monthly schedule.

Note that for fixed and floating commodity legs, the AdjustEndDateToPreviousMonthEnd field can be added to automatically adjust the end date to the end of the previous month:

`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 to the end of previous month.

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

In certain cases, a sequence of periods may be provided which do not fit within the `Rules` provided by `ScheduleData`. In this case, one may use the `Dates` node provided by `ScheduleData`. As an example of such a case, consider table 9 which shows the periods for a commodity swap leg on the arithmetic average of the nearby month NYMEX WTI future contract settlement price. In this example, the *Calculation Period* runs from the day after the previous future contract expiry to and including the nearby month's contract expiry. In this case, we need to use explicit dates as shown in listing 207.

Start Date	End Date
2019-11-21	2019-12-19
2019-12-20	2020-01-21
2020-01-22	2020-02-20
2020-02-21	2020-03-20

Table 9: Commodity derivative explicit schedule.

2.3.24 Commodity Floating Leg Data

The `CommodityFloatingLegData` node outline is shown in listing 208. The meaning and allowable values for each node are as follows:

- `Name`: An identifier specifying the commodity being referenced in the leg. Table 25 lists the allowable values for `Name` and gives a description.
- `PriceType`: It is *Spot* if the leg is referencing a commodity spot price. It is *FutureSettlement* if the leg is referencing a commodity future contract settlement price.

  Allowable values: *Spot*, *FutureSettlement*
- `Quantities`: This node is used to specify a constant quantity or a quantity that varies over the calculation periods. The usage of this node is analogous to the usage of the `Notionals` node as outlined in section 2.3.3.

  Each `Quantity` is the number of units of the underlying commodity covered by the transaction or calculation period. The unit type is defined in the underlying contract specs for the commodity name in question. For avoidance of doubt, the `Quantity` is the number of units of the underlying commodity, not the number of contracts.
- `CommodityQuantityFrequency` [Optional]: In some cases, the quantity in a commodity derivatives contract is given as a quantity per time period. This quantity is then multiplied by the number of such time periods in each calculation period to give the quantity relevant for that full calculation period. The `CommodityQuantityFrequency` can be set to
  - *PerCalculationPeriod*: This indicates that quantitie(s) as given are for the full calculation period and that no multiplication or alteration is required. This is the default setting if this node is omitted.
  - *PerPricingDay*: This indicates that the quantitie(s) are to be considered per pricing date. In general, this can be seen on averaging contracts where the quantity provided must be multiplied by the number of pricing dates in the averaging period to give the quantity applicable for the full calculation period i.e. the quantity to which the average price over the period is applied.
  - *PerHour*: This indicates that quantitie(s) are to be considered per hour. This is common in the electricity markets. The quantity then must be multiplied by the hours per day to give the quantity for a given pricing date. Also, if the contract is averaging, the resulting daily amount is multiplied by the number of pricing dates in the period to give the quantity for the full calculation period. Note that the hours per day may be specified in the `HoursPerDay` node directly. If it is omitted, it is looked up in the conventions associated with the commodity. If it is not found there and *PerHour* is used, an exception is thrown during trade building.
  - *PerCalendarDay*: This indicates that quantitie(s) are to be considered per calendar day in the period. In other words, the quantity provided is multiplied by the number of calendar days in the period to give the quantity applicable for the full calculation period.

– *PerHourAndCalendarDay*: This indicates that quantitie(s) are to be considered per hour and per calendar day in the period. In other words, the quantity provided is multiplied by the number of calendar days and number of hours per day in the period to give the quantity applicable for the full calculation period. The number of hours per period is corrected by daylight saving hours as specified in the conventions of the commodity.

Allowable values: *PerCalculationPeriod*, *PerPricingDay*, *PerHour*, *PerCalendarDay*, *PerHourAndCalendarDay*. Defaults to *PerCalculationPeriod* if omitted.

- `CommodityPayRelativeTo` [Optional]: The allowable values for this node are `CalculationPeriodStartDate`, `CalculationPeriodEndDate`, `TerminationDate`, `FutureExpiryDate`. They specify whether payment is relative to the calculation period start date, calculation period end date, leg maturity date or the future expiry date (not allowed for averaging legs) respectively. The default is `CalculationPeriodEndDate`. The payment date is then further adjusted by the payment conventions outlined in section 2.3.3 i.e. `PaymentConvention` and `PaymentLag`. If explicit payment dates are given via the `PaymentDates` node described in section 2.3.3, then those explicit payment dates are used instead and adjusted by the `PaymentCalendar` and `PaymentConvention`.

  Allowable values: *CalculationPeriodStartDate*, *CalculationPeriodEndDate*, *TerminationDate*. Defaults to *CalculationPeriodEndDate* if omitted.

- `Spreads` [Optional]: This node allows for the addition of an optional spread to the referenced commodity price in each calculation period. The usage of this node is exactly as described in section 2.3.6, except that for a Commodity leg, the Spread is not a percentage but an amount in the currency the commodity is quoted in.

  Allowable values: Each child `Spread` element can take any real number. Defaults to zero spread in each calculation period if the `Spreads` node is omitted.

- `Gearings` [Optional]: This node allows for the multiplication of the referenced commodity price in each calculation period by an optional gearing factor. The usage of this node is exactly as described in section 2.3.6. If the `Gearings` node is omitted, the gearing is one in each calculation period. Note that any spread is added to the referenced price before the gearing is applied.

- `PricingDateRule` [Optional]: The allowable values are *FutureExpiryDate* and *None*. This setting is ignored when `IsAveraged` is *true* or when `PriceType` is *Spot*. In particular, when there is no averaging and the leg is referencing a commodity future contract price, setting `PricingDateRule` to *FutureExpiryDate* ensures that the future contract price is observed on its expiry date i.e. that the *Pricing Date* is the future contract expiry date. The particular future contract being referenced is determined by the `IsInArrears` node and the `FutureMonthOffset` node. If `IsInArrears` is *true*, a base date is set as the calculation period end date. If `IsInArrears` is *false* a base date is set as the calculation period start date. The base date's month and year is then possibly moved forward by an integral number of months using the `FutureMonthOffset` node value. If this node value is zero, the base date's month and year are

unchanged. The *Pricing Date* is then the expiry date of the future contract with base date month and base date year. Setting `PricingDateRule` to *None* allows the *Pricing Date* to be determined using the `PricingCalendar` and `PricingLag` below.

Allowable values: *FutureExpiryDate*, *None*. Defaults to *FutureExpiryDate* if omitted.

- `PricingCalendar` [Optional]: This is the business day calendar used to determine pricing date(s) and in the application of the `PricingLag` if provided. If it is omitted, the calendar that has been set up for the reference commodity future contract or referenced commodity spot price will be used.
- `PricingLag` [Optional]: Any non-negative integer is allowed here. This node indicates that the *Pricing Date* is this number of business days before a given base date. The base date is the period start date if `IsInArrears` is *true* and it is the period end date if `IsInArrears` is *false*. This setting is not used when `IsAveraged` is *true*.

  Allowable values: Any non-negative integer. Defaults to zero if omitted.
- `PricingDates` [Optional]: This node is not used when `IsAveraged` is *true*. When `IsAveraged` is *false*, this node allows the *Pricing Date* in each period to be given an explicit value. If this node is included, it must contain the same number of `PricingDate` nodes as calculation periods. In general, this node is omitted but is used when the other options do not give the desired *Pricing Date* as specified in the trade's contractual terms.
- `IsAveraged` [Optional]: This node is set to *true* if the *Floating Price* is the arithmetic average of the commodity reference price over each business day in the calculation period. This node is set to *false* if there is no averaging of the underlying commodity price. Note that `IsAveraged` must be set to *true* if the `Name` given references a future contract that is averaging itself. There is more on this below.

  Allowable values: *true*, *false*. Defaults to *false* if omitted.
- `IsInArrears` [Optional]: This node is not used when `IsAveraged` is *true*. Although, if the observed underlying is averaging itself, having `IsAveraged` set to *true* would be ignored with regards this node. As noted above, this setting determines a base date from which the *Pricing Date* is determined. The base date is the period end date if `IsInArrears` is *true* and it is the period start date if `IsInArrears` is `false`. How the *Pricing Date* is then determined from this base date is determined by the `PricingDateRule` node or the `PricingCalendar` and `PricingLag` nodes.

  Allowable values: *true*, *false*. Defaults to *true* if omitted.
- `FutureMonthOffset` [Optional]: This node allows any non-negative integer value. If this node is omitted, it is set to zero. The node has a different usage depending on whether `IsAveraged` is *true* or *false*:
  - If `IsAveraged` is *true*, this node indicates which future contract is being referenced on each *Pricing Date* in the calculation 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`.

  - If `IsAveraged` is *false*, this node acts as an offset for the contract month and is used in conjunction with the `IsInArrears` setting to determine the future contract being referenced. If `IsInArrears` is `true`, a base date is set as the calculation period end date. If `IsInArrears` is *false*, a base date is set as the calculation period start date. If `FutureMonthOffset` is zero, the future contract month and year is taken as the base date's month and year. If `FutureMonthOffset` is one, the future contract month and year is taken as the month following the base date's month and year and so on for all positive values of `FutureMonthOffset`.

- `DeliveryRollDays` [Optional]: This node allows any non-negative integer value and is only applicable when `IsAveraged` is *true*. When averaging a commodity future contract price during a calculation period, where the calculation period includes the contract expiry date, this node's value indicates when we should begin using the next future contract prices in the averaging. If the value is zero, we should include the 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 contract prices thereafter. Similarly for other non-negative integer values.

  Allowable values: Any non-negative integer. Defaults to zero if omitted.

- `IncludePeriodEnd` [Optional]: If this node is set to *true*, the period end date is included in the calculation period. If it is set to *false*, the period end date is excluded from the calculation period. There is more about this in the section 2.3.23. If this node is omitted, it is set to *true*. In general, this node should be omitted and allowed to take its default value.

- `ExcludePeriodStart` [Optional]: If this node is set to *true*, the period start date is excluded from the calculation period. If it is set to *false*, the period start date is included from the calculation period. There is more about this in the section 2.3.23. If this node is omitted, it is set to *true*. In general, this node should be omitted and allowed to take its default value.

- `HoursPerDay` [Optional]: This node is used if `CommodityQuantityFrequency` is set to *PerHour* or *PerHourAndCalendarDay*. It is described above under `CommodityQuantityFrequency`.

  Allowable values: A number between 0 and 24. If omitted it defaults to the value of the `HoursPerDay` node in the conventions for the referenced commodity.

- `UseBusinessDays` [Optional]: A boolean flag that defaults to *true* if omitted. It is not applicable if `IsAveraged` is *false*. 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` to generate the correct pricing dates in the averaging period.

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

- `UnrealisedQuantity` [Optional]: A boolean flag that defaults to *false* if omitted. This is a rarely used flag. When set to *true*, it allows the user, on a given valuation date, to enter the current period quantity as an amount remaining in the current period after the valuation date i.e. the unrealised portion of the current period's quantity. This unrealised quantity is then scaled up internally to give the quantity over the full period.

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

- `LastNDays` [Optional]: This node allows a positive integer value less than or equal to 31 and is currently only supported when `PriceType` is `FutureSettlement`. When included, instead of the commodity future price being observed on the single *Pricing Date* in the period, it is observed on the `LastNDays` *Pricing Date*s, up to and including the original *Pricing Date*, for which future settlement prices are available.

- `Tag` [Optional]: This node takes any string and can be used to link the floating leg with a fixed leg that has not explicitly provided its own quantities. This can be useful in situations where the quantities on the floating leg are specified with a `CommodityQuantityFrequency` that is not simply `PerCalculationPeriod`. The fixed leg does not have the `CommodityQuantityFrequency` field. In these cases, the fixed leg can omit its `Quantities` node and take the quantities from the floating leg. This `Tag` node allows the fixed leg to link to a specific floating leg if there is more than one floating leg on the trade i.e. the fixed leg must just have the same `Tag`. The link is also used to set the payment dates of the fixed leg if CommodityPayRelativeTo is set to FutureExpiryDate.

- `DailyExpiryOffset` [Optional]: This node allows any non-negative integer value. It only has effect the underlying commodity `Name` is not being averaged and has a daily contract frequency.

  If this node is omitted, it defaults to zero. This node indicates which future contract is being referenced on each *Pricing Date* by acting as a business day offset, using the commodity `Name`'s expiry calendar, from the *Pricing Date*. It is useful e.g. in the base metals market where a future contract 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*. In this case, the `DailyExpiryOffset` would be set to `2`.

- `FXIndex` [Optional]: If `IsAveraged` is *true* this node allows the fx conversion to be applied daily in the computation of averaged cash flows. It cannot be used

with the `Indexing` node.

Allowable values: See Table 21 for supported fx indices.

- `AvgPricePrecision` [Optional]: This is only applicable when averaging is enabled. Allowed values: non-negative integer. Specifies the number of decimal places to which the average price should be rounded.

We note above that `IsAveraged` must be set to *true* if the `Name` given references a future contract that is averaging itself. For the avoidance of doubt, this does not lead to the prices of the averaging future contract being averaged in each calculation period. Instead, a check is performed in the code if the contract defined by `Name` is averaging, and if the leg itself is averaging we switch to observing the averaging future contract price on the single *Pricing Date* determined by the `PricingDateRule` node or the `PricingCalendar` and `PricingLag` nodes or the `PricingDates` node. This is best illustrated using an example. Suppose that we have a commodity swap with the schedule shown in table 10. Suppose that the *Floating Price* for the swap is specified as *For each Calculation Period, the arithmetic average of the Commodity Reference Price, for each Commodity Business Day in the Calculation Period* and that the *Commodity Reference Price* is specified as *OIL-WTI-NYMEX* with *Delivery Date* of *First Nearby Month*. There are two approaches to setting up the XML for this commodity floating leg:

1. The first approach is shown in listing 209. Note that the `Name` is `NYMEX:CL` to indicate the NYMEX WTI future contract, `IsAveraged` is `true` and `FutureMonthOffset` is `0` to indicate that we are using the nearby month contract price in the averaging. This approach is clear.

2. The second approach is to use the `CommodityFloatingLegData` shown in listing 210. Note that we have changed the `Name` to `NYMEX:CSX` to reference the NYMEX WTI Financial Futures contract. This future contract settlement price at expiry is the exact payoff of the swap leg in that it is the arithmetic average of the nearby month NYMEX WTI future contract settlement prices over the calendar month. The contract details are given here. We keep `IsAveraged` set to `true`. If we set `IsAveraged` to `false`, an error will be thrown. When `IsAveraged` is set to `true` and the `Name` references a future contract that is averaging, it is understood that the commodity leg is to use the same averaging as the future contract. In this case, we switch to a non-averaging cashflow in the code and read the averaged price directly off the price curve that we have set up using the averaging future contract prices.

In some cases, we will only have an averaging future contract available as an allowable `Name` value. For example, `NYMEX:A7Q` is one such instance. The contract details are given here. This future contract's price at the end of each contract month is the *arithmetic average of the OPIS Mt. Belvieu Natural Gasoline (non-LDH) price for each business day during the contract month*. The corresponding commodity floating leg would be set up with `Name` set to `NYMEX:A7Q` and `IsAveraged` set to `true`. Again, for the avoidance of doubt, we are not averaging the averaging future contract price. Instead, we switch to a non-averaging cashflow in the code and read the averaged price directly off the price curve that we have built out of `NYMEX:A7Q` future contract prices. We are pricing a leg that has the same payoff as the future contract.

If we have an averaging coupon and the valuation date is during the coupon period, the choice between the first and second approach above will have an effect on the sensitivities that are generated for that one single coupon. It should not affect the NPV of the coupon. The effect becomes more pronounced as the number of days remaining in the coupon period reduce. In the first approach, the coupon is priced by reading the expected future prices on future *Pricing Date*s off the non-averaging future price curve and fetching past fixed settlement prices on past *Pricing Date*s. All of these prices are then averaged. It is clear that as the valuation date approaches the final date in the coupon period, the sensitivity decreases because any bump in the curve used for pricing is only affecting the values on the remaining future *Pricing Date*s. In the second approach, the average price relevant for the full coupon period is read directly off the averaging future price curve. Any bump to the averaging future price curve affects the full coupon regardless of the position of the valuation date in the coupon period. The sensitivity will therefore be larger than using the first approach and the difference will become more noticeable as the valuation date moves towards the end of the coupon period. This subtlety can lead to differences that are larger than expected on basis swaps with averaging coupons and short maturities. If one commodity floating leg references a non-averaging price curve and the other leg references an averaging price curve, the differing effects of the bump outlined above on each leg can lead to a larger than expected net sensitivity.

Start Date	End Date	Quantity Per Period
2019-09-01	2019-09-30	5,000
2019-10-01	2019-10-31	5,000

Table 10: Example commodity swap schedule.

2.3.25 Equity Margin Leg

An equity margin leg is specified in a `LegData` node with `LegType` set to `EquityMargin`. It is used to define a sequence of cashflows that are linked to an equity price and it's associated margin factor. Each cashflow has an associated *Calculation Period*. This leg is typically used to represent a part of a Total Return Swap (TRS) on an Equity Index Future. The full TRS on the Equity Index Future uses TradeType *Swap*, and one leg of type *Equity*, and the other leg of type *EquityMargin*. Note that the equity identifier on both legs (the Name field) should be for the Equity Index, and not the Future.

The outline of a equity margin leg is given in listing 211. It has the usual `LegData` elements described in section 2.3.3 and a `EquityMarginLegData` node that is described in section 2.3.26 below.

2.3.26 Equity Margin Leg Data

The `EquityMarginLegData` node outline is shown in listing 211. The meaning and allowable values for each node are as follows:

- `Rates`: The fixed real rate(s) of the leg. While this can be a single value, a vector of values or a dated vector of values. Allowable values: Each rate element can take any real number. The rate is expressed in decimal form, e.g. *0.05* is a rate of 5%..

- `InitialMarginFactor`: this node is used to specify the equity margin factor for the first period of the trade. It's a percentage that reflecting the current applicable official Exchange initial margin requirement. It is expressed in decimal form, e.g. *0.05* is a rate of 5%..
- `EquityLegData`: this node is used to specify the underlying equity details. It's values are as outlined in section 2.3.16.
- `Multiplier` [Optional]: in some cases, the cashflow amounts are multiplied by a fixed amount. Defaults to 1.

2.3.27 CDS Reference Information

This trade component can be used to define the reference entity, tier, currency and documentation clause in credit derivative trades. For example, it can be used in the `CreditDefaultSwapData` section in a CDS trade and in the `BasketData` section in credit derivatives involving more than one underlying reference entity. The value for each of these fields is generally agreed and specified in the credit derivative contract and they determine the credit curve that is used in pricing the trade.

Listing 213: CDS reference information node

```
<ReferenceInformation>
  <ReferenceEntityId>...</ReferenceEntityId>
  <Tier>...</Tier>
  <Currency>...</Currency>
  <DocClause>...</DocClause>
</ReferenceInformation>
```

The meanings and allowable values of the elements in the `ReferenceInformation` node are as follows:

- `ReferenceEntityId`: This is typically a six digit Markit RED code specifying the underlying reference entity with the prefix `RED:` e.g. `RED:008CA0`.
- `Tier`: The debt tier that is applicable for the specified reference entity in the credit derivative. Table 26 provides the allowable values.
- `Currency`: The currency that is applicable for the specified reference entity in the credit derivative. Table 15 provides the allowable values.
- `DocClause`: The documentation clause that is applicable for the specified reference entity in the credit derivative. This defines what constitutes a credit event for the contract as well as any limitations on the deliverable debt in the event of a credit event. Table 27 provides the allowable values.

2.3.28 Basket Data

This trade component node is used in credit derivative trades referencing more than one reference entity e.g. in the `IndexCreditDefaultSwapData` node of an index CDS trade. It contains `Name` sub-nodes with the details of each constituent reference entities (names) of the basket. An example structure of the `BasketData` trade component node is shown in Listing 214.

Listing 214: Basket data

```
<BasketData>
  <Name>
    <IssuerId>CPTY_1</IssuerId>
    <CreditCurveId>RED:...</CreditCurveId>
    <Notional>100000.0</Notional>
    <Currency>USD</Currency>
  </Name>
  <Name>
    <IssuerId>CPTY_2</IssuerId>
    <CreditCurveId>RED:...</CreditCurveId>
    <Notional>100000.0</Notional>
    <Currency>USD</Currency>
  </Name>
  <Name>
    <IssuerId>CPTY_3</IssuerId>
    <CreditCurveId>RED:...</CreditCurveId>
    <Notional>100000.0</Notional>
    <Currency>USD</Currency>
  </Name>
  ...
</BasketData>
```

The meanings and allowable values of the elements in each `Name` sub-node of the `BasketData` node follow below.

- IssuerId: A unique identifier for the index component reference entity. For informational purposes and not used for pricing.

  Allowable values: Any alphanumeric string.

- CreditCurveId: The unique identifier of the index component defining one of the default curves used for pricing. The pricing can be set up to either use the curve identifiers of the index components, or one single index curve id defined in the trade specific data. A `ReferenceInformation` node may be used in place of this `CreditCurveId` node.

  Allowable values: See `CreditCurveId` for credit trades - single name in Table 23. Duplicate CreditCurveId:s are not allowed.

- `ReferenceInformation`: This node may be used as an alternative to the `CreditCurveId` node to specify the reference entity, tier, currency and documentation clause for the basket constituent. This in turn defines the credit curve used for this basket constituent in the pricing. The `ReferenceInformation` node is described in further detail in Section 2.3.27.

- Notional: The notional of the index component reference entity. Note that the sum of index component notionals (all names) must match the fixed premium leg notional. Allowable values: Any positive real number.

- Weight: Can be used, instead of Notional, to specify the weight of the index component reference entity. Note that the sum of index component notionals (all names) must match 1. Allowable values: Any positive real number.

- Currency: Defines the currency of the component, only mandatory together with a given notional.

2.3.29 Underlying

This trade component can be used to define the underlying entity for an Equity, Commodity or FX trade, but it can also define an underlying interest rate, inflation index, credit name or an underlying bond. It can be used for a single underlying, or within a basket with associated weight. For an equity underlying a string representation is used to match `Underlying` node to required configuration and reference data. The string representation is of the form IdentifierType:Name:Currency:Exchange, with all entries optional except for Name.

Listing 215: Underlying node

```
<Underlying>
  <Type>...</Type>
  <Name>...</Name>
  <Weight>...</Weight>
  <Currency>...</Currency>
  <IdentifierType>...</IdentifierType>
  <Exchange>...</Exchange>
  <PriceType>...</PriceType>
  <FutureMonthOffset>...</FutureMonthOffset>
  <DeliveryRollDays>...</DeliveryRollDays>
  <DeliveryRollCalendar>...</DeliveryRollCalendar>
</Underlying>
```

Example structures of the `Underlying` trade component node are shown in Listings 216 and 217 for an equity underlying, in Listing 220 for an fx underlying, in Listing 221 for a commodity underlying, in Listing 222 for an underlying interest rate index, in Listing 223 for an underlying inflation index, in Listing 224 for an underlying credit name, in listing 225 for an underlying bond.

Listing 216: Equity Underlying - RIC

```
        <Underlying>
            <Type>Equity</Type>
            <Name>.SPX</Name>
            <Weight>1.0</Weight>
            <IdentifierType>RIC</IdentifierType>
        </Underlying>
```

Listing 217: Equity Underlying - ISIN

```
<Underlying>
    <Type>Equity</Type>
    <Name>NL0000852580</Name>
    <Weight>1.0</Weight>
    <IdentifierType>ISIN</IdentifierType>
    <Currency>EUR</Currency>
    <Exchange>XAMS</Exchange>
</Underlying>
```

Listing 218: Equity Underlying - FIGI

```
<Underlying>
    <Type>Equity</Type>
    <Name>BBG000BLNNV0</Name>
    <IdentifierType>FIGI</IdentifierType>
</Underlying>
```

Listing 219: Equity Underlying - Bloomberg Identifier (Parsekey)

```
<Underlying>
    <Type>Equity</Type>
    <Name>BARC LN Equity</Name>
    <IdentifierType>BBG</IdentifierType>
</Underlying>
```

Listing 220: FX Underlying

```
<Underlying>
  <Type>FX</Type>
  <Name>ECB-EUR-USD</Name>
  <Weight>1.0</Weight>
</Underlying>
```

Listing 221: Commodity Underlying

```
<Underlying>
  <Type>Commodity</Type>
  <Name>NYMEX:CL</Name>
  <Weight>1.0</Weight>
  <PriceType>FutureSettlement</PriceType>
  <FutureMonthOffset>0</FutureMonthOffset>
  <DeliveryRollDays>0</DeliveryRollDays>
  <DeliveryRollCalendar>TARGET</DeliveryRollCalendar>
  <FutureContractMonth>Nov2023</FutureContractMonth>
</Underlying>
```

Listing 222: InterestRate Underlying

```
        <Underlying>
          <Type>InterestRate</Type>
          <Name>USD-CMS-10Y</Name>
          <Weight>1.0</Weight>
        </Underlying>
```

Listing 223: Inflation Index Underlying

```
        <Underlying>
          <Type>Inflation</Type>
          <Name>USCPI</Name>
          <Weight>1.0</Weight>
          <!-- optional -->
          <Interpolation>Linear</Interpolation>
</Underlying>
```

Listing 224: Credit Underlying

```
        <Underlying>
          <Type>Credit</Type>
          <Name>ISSUER_A</Name>
          <Weight>1.0</Weight>
        </Underlying>
```

Listing 225: Bond Underlying

```
      <Underlying>
        <Type>Bond</Type>
        <Name>US69007TAB08</Name>
        <IdentifierType>ISIN</IdentifierType>
        <Weight>0.5</Weight>
        <BidAskAdjustment>-0.0025</BidAskAdjustment>
      </Underlying>
```

The meanings and allowable values of the elements in the `Underlying` node are as follows:

- `Type`: The type of the Underlying asset.

  Allowable values: *Equity*, *FX*, *Commodity*, *InterestRate*, *Inflation*, *Credit*, *Bond*

- `Name`: The name of the Underlying asset.

  Allowable values:

  Equity: See `Name` for equity trades in Table 24

  FX: A string on the form SOURCE-CCY1-CCY2, where SOURCE is the FX fixing source, and the fixing is expressed as amount in CCY2 per one unit of

CCY1. See Table 21, and note that the FX- prefix is not included in `Name` as it is already included in `Type`.

InterestRate: Any valid interest rate index name, see Table 19

Inflation: Any valid zero coupon inflation index (CPI) name, See Table 22

Credit: Any valid credit name with a configured default curve, see Table 23

Bond: Any valid bond identifier, the bond must be set up in the reference data.

Commodity: An identifier specifying the commodity being referenced in the leg. Table 25 lists the allowable values for `Name` and gives a description.

- `Weight` [Optional]: The relative weight of the underlying if part of a basket. For a single underlying this can be omitted or set to 1.

  Allowable values: A real number. Defaults to 1 if left blank or omitted. A value of zero means that the underlying is excluded from the basket.

  Notes on negative weights in the *TotalReturnSwap* trade type:

  Negative weights for EquityOptionPositions are allowed, but not recommended. A negative weight for an EquityOptionPosition is equivalent to inverting the LongShort flag in the respective OptionData node.

  For EquityPositions a negative weight means that flows are in the opposite direction of the Payer flag on the return leg. A use case for negative weights is for a basket of EquityPositions that include both long and short positions.

- `IdentifierType` [Optional]: Only valid when `Type` is *Equity* or *Bond*. The type of the identifier being used.

  Allowable values: *RIC*, *ISIN*, *FIGI*, *BBG*. Defaults to *RIC*, if left blank or omitted, and `Type`: is *Equity*.

- `Currency` [Mandatory when `IdentifierType` is *ISIN*]: Only valid when `Type` is *Equity*. The currency the underlying equity is quoted in. Used when `IdentifierType` is *ISIN*, to - together with the `Exchange` convert a given ISIN to a RIC code.

  Allowable values: See Table 15 `Currency`. Mandatory when `IdentifierType` is *ISIN*, and should not be used for other `IdentifierType`:s When `Type` is *Equity*, Minor Currencies in Table 15 are also allowable.

- `Exchange` [Mandatory when `IdentifierType` is *ISIN*]: Only valid when `Type` is *Equity*. A string code representing the exchange the equity is traded on. Used when `IdentifierType` is *ISIN*, to - together with the `Currency` convert a given ISIN to a RIC code.

  Allowable values: The MIC code of the exchange, see Table 28. Mandatory when `IdentifierType` is *ISIN*, and should not be used for other `IdentifierType`:s.

- `PriceType` [Optional]: Only valid when `Type` is *Commodity*. Whether the Spot or Future price is referenced.

  Allowable values: *Spot*, *FutureSettlement*. Mandatory when `Type` is *Commodity* .

- `FutureMonthOffset` [Optional]: Only valid when `Type` is *Commodity*. Only relevant for the *FutureSettlement* price type, in which case the $N + 1$th future with expiry greater than ObservationDate for the given commodity underlying will be referenced.

  Allowable values: An integer. Mandatory for when `Type` is *Commodity* and `PriceType` is *FutureSettlement*.

- `DeliveryRollDays` [Optional]: Only valid when `Type` is *Commodity*. The number of days the observation date is rolled forward before the next future expiry is looked up.

  Allowable values: An integer. Defaults to 0 if left blank or omitted, and `Type`: is *Commodity*.

- `DeliveryRollCalendar` [Optional]: Only valid when `Type` is *Commodity*. The calendar used to roll forward the observation date.

  Allowable values: See Table 17. Defaults to the null calendar if left blank or omitted, and `Type`: is *Commodity*.

- `FutureContractMonth` [Optional]: Only valid when `Type` is *Commodity*, `PriceType` is FutureSettlement and there is no `FutureExpiryDate` node. It specifies the underlying future contract month in the format *MonYYYY*, for example Nov2023.

- `FutureExpiryDate` [Optional]: Only valid when `Type` is *Commodity*, `PriceType` is FutureSettlement and there is no `FutureContractMonth` node. This gives the expiration date of the underlying commodity future contract.

  If the field `FutureExpiryDate` and `FutureContractMonth` are omitted, the expiration date of the underlying commodity future contract is set to the prompt future, adjusted for any `FutureMonthOffset`.

- `Interpolation` [Optional]: Only valid when `Type` is *Inflation*. The index observation interpolation between fixings.

  Allowable values: Flat, Linear

- `BidAskAdjustment` [Optional]: Only valid when `Type` is *Bond*. A correction applied to the price found in the market data (usually mid), if the bond basket price is defined on the bid or ask side rather than mid.

  Allowable values: Any real number.

2.3.30 StrikeData

This trade component that can be used to define the strike entity for commodity, equity and bond options. It can be used to define either a Price or Yield strike, with examples below in 226 and 227 respectively.

Listing 226: Strike Price

```
<StrikeData>
        <StrikePrice>
                <Value>1</Value>
                <Currency>EUR</Currency>
        </StrikePrice>
</StrikeData>
```

The meanings and allowable values of the elements in the `StrikePrice` node are as follows:

- `Value`: The strike price.

  Allowable values: Any positive real number.

- `Currency` [Mandatory for Quanto/Compo, Optional otherwise]: The currency of the amount given in `Value`, i.e. the strike currency.

  Note:
  Quanto: The payment/leg currency and the currency the underlying asset is quoted in differ. The strike currency is in the currency the asset is quoted in.
  Compo (Composite): The payment/leg currency and the currency the underlying asset is quoted in differ. The strike currency is in the payment/leg currency.

  Allowable values: See Table 15 `Currency`. Minor Currencies in Table 15 are also allowable. In non-quanto/compo cases, if left blank or omitted, it defaults to the currency of the leg for equity and commodity options, and to the currency the underlying bond is quoted in for BondOptions using reference data.

Listing 227: Strike Yield

```
<StrikeData>
        <StrikeYield>
                <Yield>0.055</Yield>
                <Compounding>SimpleThenCompounded</Compounding>
        </StrikeYield>
</StrikeData>
```

The meanings and allowable values of the elements in the `StrikeYield` node are as follows:

- `Yield`: A Yield quoted in decimal form, e.g. 10% should be entered as 0.1.

  Allowable values: Any real number.

- `Compounding` [Optional]: The compounding or the yield given in `Yield`.

  Allowable values: *Simple, Compounded, Continuous, SimpleThenCompounded.* Defaults to *SimpleThenCompounded* if left blank or omitted.

Trade Data Container	Supported Barrier Styles
FxBarrierOptionData	*American*
FxDigitalBarrierOptionData	*American*
FxEuropeanBarrierOptionData	*European*
FxTouchOptionData	*American*
FxDoubleTouchOptionData	*American*
FxDoubleBarrierOptionData	*American*
FxKIKOBarrierOptionData	*American*
FxTaRFData	*European*
FxAccumulatorData	*European, American*
EquityTaRFData	*European*
EquityAccumulatorData	*European, American*
CommodityAccumulatorData	*European, American*
FxGenericBarrierOption	*American*
EquityGenericBarrierOption	*American*
CommodityGenericBarrierOption	*American*

Table 11: Supported barrier styles per trade data container

2.3.31 Barrier Data

This trade component node is used within the trade data containers listed in table 11. Note that not every trade type allows for all barrier styles, the allowable combinations are listed in in table 11.

The barrier data element is specified as in listing 228

Listing 228: Barrier data

```
<BarrierData>
    <Type>UpAndIn</Type>
    <Style>American</Style>
    <Levels>
        <Level>1.2</Level>
    </Levels>
    <Rebate>100000</Rebate>
    <RebateCurrency>USD</RebateCurrency>
    <RebatePayTime>atExpiry</RebatePayTime>
    <OverrideTriggered>true</OverrideTriggered>
</BarrierData>
```

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

- Type: Specifies barrier type. The allowable values are given in Table 12.

Type	Description
UpAndOut	The underlying price starts below the barrier level and has to move up for the option to be knocked out.
DownAndOut	The underlying price starts above the barrier level and has to move down for the option to become knocked out.
UpAndIn	The underlying price starts below the barrier level and has to move up for the option to become activated.
DownAndIn	The underlying price starts above the barrier level and has to move down for the option to become activated.
KnockOut	For double level only. The underlying price starts between the barrier levels and has to move up or down for the option to be knocked out.
KnockIn	For double level only. The underlying price starts between the barrier levels and has to move up or down for the option to become activated.
CumulatedProfitCap	For TaRFs only. The instrument terminates once the generated profit reaches the CumulatedProfitCap.
CumulatedProfitCapPoints	For TaRFs only. The instrument terminates once the generated profit divided by fixing amount and absolute value of leverage reaches the CumulatedProfitCapPoints.
FixingCap	For TaRFs only. The instrument terminates once the number of observations where a profit is generated reaches the FixingCap.
FixingFloor	For Accumulators only. The first n fixings are guaranteed regardless of whether the trade has been knocked out already.

Table 12: Allowable Type Values.

- Style[Optional]: Specifies the monitoring style of the barrier. Optional, if not given, defaults to the supported barrier style (see table 11 and if both *American* and *European* barriers are supported, defaults to *American*.
  Allowable values: *American, European.*
- Level: The barrier level, defined as the amount in sold (domestic) currency per unit bought (foreign) currency. Double barrier instruments can have two `Level` elements, and these must be in ascending order.
  Allowable values: Any positive real number.
- Rebate[Optional]: The barrier rebate is a fixed amount, expressed in domestic / sold currency paid out to the option holder if a barrier option expires inactive, i.e. it is not knocked in/out. Note that `Rebate` is supported for
  - FxBarrierOptionData
  - FxDigitalBarrierOptionData
  - FxDoubleBarrierOptionData
  - FxEuropeanBarrierOptionData
  - FxGenericBarrierOptionData
  - EquityGenericBarrierOptionData

– CommodityGenericBarrierOptionData

only. If defined for several "in" barriers, the amounts must be identical across all barrier definitions (because the rebate amount is paid if none of the "in" barrier is touched and can therefore not depend on the particular barrier). Also, the RebatePayTime must be *atExpiry* for "in" barriers obviously.
Allowable values: Any positive real number. Defaults to zero if omitted. Cannot be left blank.

- RebateCurrency [Optional]: The currency in which the rebate amount is paid. Defaults to the natural pay currency of the trade. Deviating currencies are supported by the following trade types only:
  - FxGenericBarrierOptionData
  - EquityGenericBarrierOptionData
  - CommodityGenericBarrierOptionData

  Allowable Values: See Table 15 `Currency`.
- RebatePayTime [Optional]: For "in" barriers only atExpiry is allowed. For "out" barriers, both atExpiry and atHit is possible. If not given, defaults to "atExpiry". This field is only supported by the following trade types:
  - FxGenericBarrierOptionData
  - EquityGenericBarrierOptionData
  - CommodityGenericBarrierOptionData

  Allowable Values: *atExpiry*, *atHit*
- StrictComparison [Optional]: Determines how the barrier is checked, as per:

  0: the barrier checks use <=, >= to check In-barriers and <, > to check Out-barriers.

  1: the barrier checks use strict comparison < and > for both In- and Out-barriers.

  2: the barrier checks use strict or equal comparison <= and >= for both In- and Out-barriers.

  Note that the StrictComparison element only has an effect for TradeType *GenericBarrierOption*. All other TradeTypes are treated as having StrictComparison set to *0*.

  Allowable Values: *0*, *1*, or *2*. Defaults to *0* if omitted.
- OverrideTriggered [Optional]: Specifies whether a barrier was triggered before the valuation date. If given, this overrides the automatic check using fixing data.

  Allowable Values: *true*, *false*

2.3.32 RangeBound

This trade component node is used within the following trade data containers

- FxTaRFfData, EquityTaRFData, CommodityTaRFData
- FxAccumulatorData, EquityAccumulatorData, CommodityAccumulatorData

An example structure of the `RangeBound` trade component node is shown in Listing 229.

Listing 229: RangeBound

```
<RangeBound>
  <RangeFrom>0</RangeFrom>
  <RangeTo>155.00</RangeTo>
  <Leverage>2</Leverage>
  <Strike>150.54</Strike>
</RangeBound>
```

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

- RangeFrom [Optional]: The lower bound of the range.

  Allowable values: Any real number. If omitted, no lower bound applies. Cannot be left blank.

- RangeTo [Optional]: The upper bound of the range.

  Allowable values: Any real number. If omitted, no lower bound applies. Cannot be left blank.

- Leverage [Optional]: The leverage that applies to the range. For TaRFs, negative leverage can be mixed with positive leverage to reflect a TaRF with switching buyer/seller. However, for Accumulators all given Leverage parameters within the same instrument (in multiple `RangeBound` nodes) must have the same sign.

  Allowable values: Any real number. Defaults to 1 if omitted. Cannot be left blank.

- Strike [Optional]: The strike specific to the range. If given overwrites a strike given on the trade level.

  Allowable values: Any real number. Defaults to the trade level strike if omitted. Cannot be left blank.

- StrikeAdjustment [Optional]: A strike adjustment relative to the strike given on the trade level. If given the strike for the defined range is computed as $K + A$ where K is the strike on the trade level and A is the strike adjustment. Notice that Strike and StrikeAdjustment can not be given both at the same time.

  Allowable values: Any real number.

2.3.33 Bond Basket Data for Cashflow CDO

This trade component node is used in a Cashflow CDO trade as explained in 2.2.50. An example structure of the `BondBasketData` trade component node is shown in Listing 230.

Listing 230: Bond Basket Data for Cashflow CDO

```
<BondBasketData>
    <Trade id="Bond_1">
      <TradeType>Bond</TradeType>
      <Envelope>
        ...
      </Envelope>
      <BondData>
        ...
      </BondData>
    </Trade>
    <Trade id="Bond_2">
      <TradeType>Bond</TradeType>
      <Envelope>
        ...
      </Envelope>
      <BondData>
        ...
      </BondData>
    </Trade>
</BondBasketData>
```

The usage of the BondBasketData is akin to a portfolio of bond trades, but is embraced by the keyword `BondBasketData` as opposed to `Portfolio`. Compare the vanilla bond section 2.2.40 for usage and allowable values.

2.3.34 CBO Tranches

This trade component node is used in a CBO trade as explained in 2.2.50. An example structure of the `CBOTranches` trade component node is shown in Listing 231.

Listing 231: CBO Tranches

```
<CBOTranches>
  <Tranche>
    <Name>JuniorNote</Name>
    <ICRatio>0.0</ICRatio>
    <OCRatio>0.0</OCRatio>
    <Notional>4000000.00</Notional>
    <FixedLegData>
      <Rates>
        <Rate>0.03</Rate>
      </Rates>
    </FixedLegData>
  </Tranche>
  ...
</CBOTranches>
```

The meanings of the elements of the `CBO tranches` node follow below:

- Tranche: Multiple tranches are allowed and are indicated by the tranche node within the embracing CBOTranches node.

- Name: This string is the name of the tranche, possibly reflecting the position in the capital structure.
- ICRatio: The interest coverage ratio is a number, defined as BasketInterest over TrancheInterest (incl. all senior tranches).
- OCRatio: The overcollateralisation ratio is a number, defined as BasketNotional over TrancheNotional (incl. all senior tranches).
- Notional: The face amount of the tranche.

Depending on the tranche, one can specify a floating or fixed return via the nodes:

- FixedLegData, which is outlined in section 2.3.5.
- FloatingLegData, which is outlined in section 2.3.6.

2.3.35 Formula Based Leg Data

The formula based leg data allows to use complex formulas to describe coupon payoffs. Its `LegType` is `FormulaBased`, and it has the data section `FormulaBasedLegData`. It supports IBOR and CMS based payoffs with quanto and digital features. The following example shows the definition of a coupon paying a capped / floored cross currency EUR-GBP CMS Spread contingent on a USD CMS barrier.

The `Index` field supports operations of the following kind:

- indices like IBOR and CMS indices, and constants as factors, spreads and/or cap/floor values;
- basic operations: $+$, $-$, $\cdot$, $/$;
- operators gtZero() (greater than zero) and geqZero() (greater than or equal zero) yielding 1 if the argument is > 0 (resp. ≥ 0) and zero otherwise
- functions: abs(), exp(), log(), min(), max(), pow()

In listing 232, we present a `FormulaBasedLegData` example.

This leg data type can be used in Swap and Bond trades.

Listing 131: Payoff script for a Worst Of Basket Swap.

```
REQUIRE SIZE(Underlyings) == SIZE(InitialPrices);
REQUIRE SIZE(SettlementDates) == SIZE(DeterminationDates);
REQUIRE SIZE(KnockOutLevels) == SIZE(DeterminationDates) - 1;
REQUIRE SIZE(CouponTriggerLevels) == SIZE(DeterminationDates) - 1;

NUMBER alive, couponAccumulation, numOfKnockedAssets, fixing, n, accrualFraction, indexInitial;
NUMBER numOfTriggeredAssets, indexFinal, performance, worstPerformance, d, payoff, u;

Option = Option + LOGPAY(LongShort * Quantity * InitialFixedRate,
                         SettlementDates[1], SettlementDates[1], PayCcy, 0, InitialFixedAmount);

alive = 1;
couponAccumulation = 1;
n = SIZE(DeterminationDates);

FOR d IN (2, n, 1) DO
  fixing = FloatingIndex(FixingSchedule[d-1]) + FloatingSpread;
  accrualFraction = dcf(FloatingDayCountFraction, DeterminationDates[d-1], DeterminationDates[d]);
  Option = Option - LOGPAY(LongShort * Quantity * alive * fixing * accrualFraction,
                           FixingSchedule[d-1], SettlementDates[d], PayCcy, 1, FloatingLeg);

  numOfTriggeredAssets = 0;
  FOR u IN (1, SIZE(Underlyings), 1) DO
    IF Underlyings[u](DeterminationDates[d]) >= CouponTriggerLevels[d-1] * InitialPrices[u] THEN
      numOfTriggeredAssets = numOfTriggeredAssets + 1;
    END;
  END;
  IF numOfTriggeredAssets == SIZE(Underlyings) THEN
    Option = Option + LOGPAY(LongShort * Quantity * alive * CouponRate * couponAccumulation,
                             SettlementDates[d], SettlementDates[d], PayCcy, 2, FixedCouponLeg);
    couponAccumulation = 1;
  ELSE
    IF AccumulatingCoupons == 1 THEN
      couponAccumulation = couponAccumulation + 1;
    END;
  END;

  IF d == n THEN
    FOR u IN (1, SIZE(Underlyings), 1) DO
      indexInitial = InitialPrices[u];
      indexFinal = Underlyings[u](DeterminationDates[n]);
      performance = indexFinal / indexInitial;

      IF {u == 1} OR {performance < worstPerformance} THEN
        worstPerformance = performance;
      END;
    END;

    IF worstPerformance < min(Strike, KnockInLevel) THEN
      payoff = worstPerformance - Strike;
      Option = Option + LOGPAY(LongShort * Quantity * alive * payoff, DeterminationDates[n],
                               SettlementDates[n], PayCcy, 3, EquityAmountPayoff);
    END;
  END;

  IF d != n THEN
    numOfKnockedAssets = 0;
    FOR u IN (1, SIZE(Underlyings), 1) DO
      IF Underlyings[u](DeterminationDates[d]) >= KnockOutLevels[d-1] * InitialPrices[u] THEN
        numOfKnockedAssets = numOfKnockedAssets + 1;
      END;
    END;
    IF numOfKnockedAssets == SIZE(Underlyings) THEN
      alive = 0;
    END;
  END;
END;
```

Listing 138: Payoff script for a TaRF.

```
REQUIRE FixingAmount > 0;
REQUIRE LongShort == 1 OR LongShort == -1;
REQUIRE SIZE(RangeUpperBounds) == SIZE(RangeLowerBounds);
REQUIRE SIZE(RangeLowerBounds) == SIZE(RangeLeverages);
REQUIRE SIZE(RangeLowerBounds) == SIZE(RangeStrikes);
REQUIRE TargetType == -1 OR TargetType == 0 OR TargetType == 1;
REQUIRE SIZE(FixingDates) == SIZE(SettlementDates);

NUMBER Payoff, d, r, PnL, wasTriggered, AccProfit, Hits, currentNotional;
NUMBER Fixing[SIZE(FixingDates)], Triggered[SIZE(FixingDates)];

FOR r IN (1, SIZE(RangeUpperBounds), 1) DO
  REQUIRE RangeLowerBounds[r] <= RangeUpperBounds[r];
  REQUIRE RangeStrikes[r] >= 0;
END;

FOR d IN (1, SIZE(FixingDates), 1) DO
  Fixing[d] = Underlying(FixingDates[d]);
  IF wasTriggered != 1 THEN
    PnL = 0;
    FOR r IN (1, SIZE(RangeUpperBounds), 1) DO
      IF Fixing[d] > RangeLowerBounds[r] AND Fixing[d] <= RangeUpperBounds[r] THEN
        PnL = PnL + RangeLeverages[r] * FixingAmount * (Fixing[d] - RangeStrikes[r]);
      END;
    END;

    IF PnL >= 0 THEN
      AccProfit = AccProfit + PnL;
      Hits = Hits + 1;
    END;

    IF {KnockOutProfitEvents > 0 AND Hits >= KnockOutProfitEvents} OR
       {KnockOutProfitAmount > 0 AND AccProfit >= KnockOutProfitAmount} THEN
      wasTriggered = 1;
      Triggered[d] = 1;
      IF TargetType == 0 THEN
        Payoff = Payoff + LOGPAY(TargetAmount - (AccProfit - PnL), FixingDates[d], SettlementDates[d
      END;
      IF TargetType == 1 THEN
        Payoff = Payoff + LOGPAY(PnL, FixingDates[d], SettlementDates[d], PayCcy, 0, Cashflow);
      END;
    ELSE
        Payoff = Payoff + LOGPAY(PnL, FixingDates[d], SettlementDates[d], PayCcy, 0, Cashflow);
    END;
  END;
END;
value = LongShort * Payoff;
currentNotional = FixingAmount * RangeStrikes[1];
```

Listing 139: Knock Out Swap

```
<Trade id="194837232">
  <TradeType>KnockOutSwap</TradeType>
  <Envelope>...</Envelope>
  <KnockOutSwapData>
    <!-- BarrierData and BarrierStartDate specify the knock out terms -->
    <BarrierData>
      <Type>UpAndOut</Type>
      <Levels>
        <Level>0.05</Level>
      </Levels>
    </BarrierData>
    <BarrierStartDate>2024-10-01</BarrierStartDate>
    <!-- we require exactly one Floating and one Fixed Leg -->
    <LegData>
      <LegType>Floating</LegType>
      ...
    </LegData>
    <LegData>
      <LegType>Fixed</LegType>
      ...
    </LegData>
  </KnockOutSwapData>
</Trade>
```

Listing 140: Payoff script for a BestOfAssetOrCashRainbowOption.

```
REQUIRE SIZE(Underlyings) == SIZE(Weights);
NUMBER u, thisPrice, bestPrice, Payoff, currentNotional;
bestPrice = Strike;
FOR u IN (1, SIZE(Underlyings)) DO
    thisPrice = Underlyings[u](Expiry) * Weights[u];
    IF thisPrice > bestPrice THEN
        bestPrice = thisPrice;
    END;
END;
Option = LongShort * Notional * PAY(bestPrice, Expiry, Settlement, PayCcy);
currentNotional = Notional * Strike;
```

Listing 141: Payoff script for a WorstOfAssetOrCashRainbowOption.

```
REQUIRE SIZE(Underlyings) == SIZE(Weights);
NUMBER u, thisPrice, worstPrice, Payoff, currentNotional;
worstPrice = Strike;
FOR u IN (1, SIZE(Underlyings)) DO
    thisPrice = Underlyings[u](Expiry) * Weights[u];
    IF thisPrice < worstPrice THEN
        worstPrice = thisPrice;
    END;
END;
Option = LongShort * Notional * PAY(worstPrice, Expiry, Settlement, PayCcy);
currentNotional = Notional * Strike;
```

Listing 142: Payoff script for a MaxRainbowOption.

```
REQUIRE SIZE(Underlyings) == SIZE(Weights);

NUMBER u, thisPrice, maxPrice, Payoff, ExerciseProbability, currentNotional;
maxPrice = 0;
FOR u IN (1, SIZE(Underlyings)) DO
    thisPrice = Underlyings[u](Expiry) * Weights[u];
    IF thisPrice > maxPrice THEN
        maxPrice = thisPrice;
    END;
END;

Payoff = max(PutCall * (maxPrice - Strike), 0);

Option = LongShort * Notional * PAY(Payoff, Expiry, Settlement, PayCcy);

IF Payoff > 0 THEN
    ExerciseProbability = 1;
END;
currentNotional = Notional * Strike;
```

Listing 143: Payoff script for a MinRainbowOption.

```
REQUIRE SIZE(Underlyings) == SIZE(Weights);
REQUIRE SIZE(Underlyings) > 0;

NUMBER u, thisPrice, minPrice, Payoff, ExerciseProbability, currentNotional;
minPrice = Underlyings[1](Expiry) * Weights[1];
FOR u IN (1, SIZE(Underlyings)) DO
    thisPrice = Underlyings[u](Expiry) * Weights[u];
    IF thisPrice < minPrice THEN
        minPrice = thisPrice;
    END;
END;

Payoff = max(PutCall * (minPrice - Strike), 0);

Option = LongShort * Notional * PAY(Payoff, Expiry, Settlement, PayCcy);

IF Payoff > 0 THEN
    ExerciseProbability = 1;
END;
currentNotional = Notional * Strike;
```

Listing 144: Payoff script for a EuropeanRainbowCallSpreadOption.

```
REQUIRE SIZE(Underlyings) == SIZE(Weights);
NUMBER perf[SIZE(Underlyings)], return, u;
FOR u IN (1, SIZE(Underlyings)) DO
  perf[u] = Underlyings[u](Expiry) / InitialStrikes[u];
END;
SORT (perf);
FOR u IN (1, SIZE(Underlyings)) DO
  return = return + Weights[u] * perf[SIZE(Underlyings) + 1 - u];
END;
Option = PAY( Notional * min( max( Floor, return - 1 ), Cap ), Expiry,
                                                  Settlement, PayCcy );
```

Listing 145: Payoff script for a RainbowCallSpreadBarrierOption.

```
REQUIRE SIZE(Underlyings) == SIZE(Weights);
REQUIRE Floor <= Cap;
NUMBER performance, perf[SIZE(Underlyings)], return, u, d, payoff, knockedIn;

FOR u IN (1, SIZE(Underlyings), 1) DO
  perf[u] = Underlyings[u](Expiry) / InitialPrices[u];
END;
SORT (perf);

FOR u IN (1, SIZE(Underlyings), 1) DO
  return = return + Weights[u] * perf[SIZE(Underlyings) + 1 - u];
END;

IF BermudanBarrier == 1 THEN
  FOR d IN (1, SIZE(BarrierSchedule), 1) DO
    IF knockedIn == 0 THEN
      FOR u IN (1, SIZE(Underlyings), 1) DO
        performance = Underlyings[u](BarrierSchedule[d]) / InitialPrices[u];
        IF performance <= BarrierLevel THEN
          knockedIn = 1;
        END;
      END;
    END;
  END;
ELSE
  FOR u IN (1, SIZE(perf), 1) DO
    IF perf[u] <= BarrierLevel THEN
      knockedIn = 1;
    END;
  END;
END;

payoff = min( max( Floor, return - Strike ), Cap );
Option = LongShort * PAY(Notional * Gearing * payoff * knockedIn,
                         Expiry, Settlement, PayCcy);
```

Listing 146: Payoff script for a AsianRainbowCallSpreadOption.

```
REQUIRE SIZE(Underlyings) == SIZE(Weights);
NUMBER perf[SIZE(Underlyings)], return, d, u;
FOR u IN (1, SIZE(Underlyings), 1) DO
  FOR d IN (1, SIZE(AveragingDates), 1) DO
    perf[u] = perf[u] + Underlyings[u](AveragingDates[d]);
  END;
  perf[u] = perf[u] / SIZE(AveragingDates);
END;
SORT (perf);
FOR u IN (1, SIZE(Underlyings), 1) DO
  return = return + Weights[u] * perf[SIZE(Underlyings) + 1 - u];
END;
Option = LongShort * PAY( Notional * min( max( Floor, return - 1 ), Cap ), Expiry,
                          Settlement, PayCcy );
```

Listing 147: Payoff script for a WorstPerformanceRainbowOption01.

```
REQUIRE SIZE(Underlyings) == SIZE(InitialPrices);
REQUIRE ObservationDate <= SettlementDate;

NUMBER u, indexInitial, indexFinal, performance;
NUMBER worstPerformance, payoff, premium;

FOR u IN (1, SIZE(Underlyings), 1) DO
  indexInitial = InitialPrices[u];
  indexFinal = Underlyings[u](ObservationDate);
  performance = indexFinal / indexInitial;

  IF {u == 1} OR {performance < worstPerformance} THEN
    worstPerformance = performance;
  END;
END;

payoff = LOGPAY(Quantity * (worstPerformance - 1), ObservationDate,
                SettlementDate, PayCcy, 1, Payoff);

IF worstPerformance < 1 THEN
  payoff = payoff * PayoffMultiplier;
END;

premium = LOGPAY(Premium, PremiumDate, PremiumDate, PayCcy, 0, Premium);

Option = LongShort * (payoff - premium);
```

Listing 148: Payoff script for a WorstPerformanceRainbowOption02.

```
REQUIRE SIZE(Underlyings) == SIZE(InitialPrices);
REQUIRE ObservationDate <= SettlementDate;
REQUIRE Floor <= 0;

NUMBER u, initialPrice, finalPrice, performance;
NUMBER worstPerformance, payoff, premium;

FOR u IN (1, SIZE(Underlyings), 1) DO
  initialPrice = InitialPrices[u];
  finalPrice = Underlyings[u](ObservationDate);
  performance = finalPrice / initialPrice;

  IF {u == 1} OR {performance < worstPerformance} THEN
    worstPerformance = performance;
  END;
END;

IF worstPerformance > 1 THEN
  payoff = PayoffMultiplier * (worstPerformance - 1);
ELSE
  IF worstPerformance < 1 THEN
    payoff = max(Floor, worstPerformance - 1);
  ELSE
    payoff = 0;
  END;
END;

payoff = Quantity * LOGPAY(payoff, ObservationDate, SettlementDate,
                           PayCcy, 1, Payoff);
premium = LOGPAY(Premium, PremiumDate, PremiumDate, PayCcy,
                 0, Premium);

Option = LongShort * (payoff - premium);
```

Listing 149: Payoff script for a WorstPerformanceRainbowOption03.

```
REQUIRE SIZE(Underlyings) == SIZE(InitialPrices);
REQUIRE ObservationDate <= SettlementDate;
REQUIRE Floor <= Cap;

NUMBER indexInitial, indexFinal, performance, d;
NUMBER worstPerformance, payoff, premium, knockedIn, u;

FOR u IN (1, SIZE(Underlyings), 1) DO
  indexInitial = InitialPrices[u];
  indexFinal = Underlyings[u](ObservationDate);
  performance = indexFinal / indexInitial;

  IF {u == 1} OR {performance < worstPerformance} THEN
    worstPerformance = performance;
  END;
END;

IF BermudanBarrier == 1 THEN
  FOR d IN (1, SIZE(BarrierSchedule), 1) DO
    IF knockedIn == 0 THEN
      FOR u IN (1, SIZE(Underlyings), 1) DO
        indexInitial = InitialPrices[u];
        indexFinal = Underlyings[u](BarrierSchedule[d]);
        performance = indexFinal / indexInitial;

        IF performance <= BarrierLevel THEN
          knockedIn = 1;
        END;
      END;
    END;
  END;
ELSE
  IF worstPerformance <= BarrierLevel THEN
    knockedIn = 1;
  END;
END;

payoff = min(Cap, max(Floor, worstPerformance - Strike));
payoff = LOGPAY(Quantity * payoff * knockedIn, ObservationDate,
                SettlementDate, PayCcy, 1, Payoff);

IF worstPerformance < 1 THEN
  payoff = payoff * PayoffMultiplier;
END;

premium = LOGPAY(Premium, PremiumDate, PremiumDate,
                 PayCcy, 0, Premium);

Option = LongShort * (payoff - premium);
```

Listing 150: Payoff script for a WorstPerformanceRainbowOption04.

```
REQUIRE SIZE(Underlyings) == SIZE(InitialPrices);
REQUIRE ObservationDate <= SettlementDate;
REQUIRE Floor <= Cap;

NUMBER indexInitial, indexFinal, performance, d;
NUMBER worstPerformance, payoff, premium, knockedIn, u;

FOR u IN (1, SIZE(Underlyings), 1) DO
  indexInitial = InitialPrices[u];
  indexFinal = Underlyings[u](ObservationDate);
  performance = indexFinal / indexInitial;

  IF {u == 1} OR {performance < worstPerformance} THEN
    worstPerformance = performance;
  END;
END;

IF BermudanBarrier == 1 THEN
  FOR d IN (1, SIZE(BarrierSchedule), 1) DO
    IF knockedIn == 0 THEN
      FOR u IN (1, SIZE(Underlyings), 1) DO
        indexInitial = InitialPrices[u];
        indexFinal = Underlyings[u](BarrierSchedule[d]);
        performance = indexFinal / indexInitial;

        IF performance <= BarrierLevel THEN
          knockedIn = 1;
        END;
      END;
    END;
  END;
ELSE
  IF worstPerformance <= BarrierLevel THEN
    knockedIn = 1;
  END;
END;

payoff = worstPerformance - Strike;
IF knockedIn == 0 THEN
  payoff = min(Cap, max(Floor, PayoffMultiplier * payoff));
END;

payoff = LOGPAY(Quantity * payoff, ObservationDate,
                SettlementDate, PayCcy, 1, Payoff);

premium = LOGPAY(Premium, PremiumDate, PremiumDate,
                 PayCcy, 0, Premium);

Option = LongShort * (payoff - premium);
```

Listing 151: Payoff script for a WorstPerformanceRainbowOption05.

```
REQUIRE SIZE(Underlyings) == SIZE(InitialPrices);
REQUIRE ObservationDate <= SettlementDate;
REQUIRE BarrierType == 1 OR BarrierType == 2;

NUMBER indexInitial, indexFinal, performance;
NUMBER worstPerformance, payoff, premium, knockedIn, u;

FOR u IN (1, SIZE(Underlyings), 1) DO
  indexInitial = InitialPrices[u];
  indexFinal = Underlyings[u](ObservationDate);
  performance = indexFinal / indexInitial;

  IF {u == 1} OR {performance < worstPerformance} THEN
    worstPerformance = performance;
  END;
END;

IF {{BarrierType == 1 OR BarrierType == 4}
      AND worstPerformance <= BarrierLevel}
OR {{BarrierType == 2 OR BarrierType == 3}
      AND worstPerformance >= BarrierLevel} THEN
  knockedIn = 1;
END;

IF knockedIn == 0 THEN
  payoff = 0;
ELSE
  payoff = max(0, PutCall * (worstPerformance - Strike));
END;

payoff = LOGPAY(Quantity * payoff, ObservationDate,
                SettlementDate, PayCcy, 1, Payoff);

premium = LOGPAY(Premium, PremiumDate, PremiumDate,
                 PayCcy, 0, Premium);

Option = LongShort * (payoff - premium);
```

Listing 152: Payoff script for a VarianceOption.

```
REQUIRE {Notional >= 0} AND {Strike >= 0};

NUMBER expectedN, realisedVariance, currPrice, currentNotional;
NUMBER prevPrice, payoff, realisedVariation, strike, premium, d;

FOR d IN (2, SIZE(ValuationSchedule), 1) DO
  currPrice = Underlying(ValuationSchedule[d]);
  prevPrice = Underlying(ValuationSchedule[d-1]);
  realisedVariance = realisedVariance + pow(ln(currPrice/prevPrice), 2);
END;

expectedN = SIZE(ValuationSchedule) - 1;
realisedVariance = (252/expectedN) * realisedVariance;

IF SquaredPayoff == 1 THEN
  realisedVariation = realisedVariance;
  currentNotional = pow(100, 2) * Notional / (2 * 100 * VarianceReference);
  strike = pow(Strike, 2);
ELSE
  realisedVariation = sqrt(realisedVariance);
  currentNotional = 100 * Notional;
  strike = Strike;
END;

payoff = currentNotional * max(PutCall * (realisedVariation - strike), 0);

NUMBER ExerciseProbability;
IF payoff > 0 THEN
  ExerciseProbability = 1;
END;

premium = PAY(PremiumAmount, PremiumDate, SettlementDate, PayCcy);
payoff = PAY(payoff, ValuationSchedule[SIZE(ValuationSchedule)],
             SettlementDate, PayCcy);
Option = LongShort * (payoff - premium);
```

Listing 153: Payoff script for a KIKOVarianceSwap.

```
REQUIRE {Notional >= 0} AND {Strike >= 0};
REQUIRE {Cap >= 0} AND {Floor >= 0};

alive = 1;
FOR d IN (2, SIZE(ValuationSchedule), 1) DO
  IF alive == 1 THEN
    currPrice = Underlying(ValuationSchedule[d]);
    prevPrice = Underlying(ValuationSchedule[d-1]);
    realisedVariance = realisedVariance + pow(ln(currPrice/prevPrice), 2);

    IF BarrierType == 3 OR BarrierType == 4 THEN
      daysBeforeKO = daysBeforeKO + 1;
    END;

    IF {BarrierType == 3 AND currPrice <= BarrierLevel} OR
       {BarrierType == 4 AND currPrice >= BarrierLevel} THEN
      alive = 0;
    END;

    IF knockedIn == 0 THEN
      IF {BarrierType == 1 AND currPrice <= BarrierLevel} OR
         {BarrierType == 2 AND currPrice >= BarrierLevel} THEN
        knockedIn = 1;
      END;
    END;
  END;
END;

expectedN = SIZE(ValuationSchedule) - 1;
realisedVariance = (252/expectedN) * realisedVariance;

IF SquaredPayoff == 1 THEN
  realisedVariation = realisedVariance;
  currentNotional = pow(100, 2) * Notional / (2 * 100 * Strike);
  strike = pow(Strike, 2);
ELSE
  realisedVariation = sqrt(realisedVariance);
  currentNotional = 100 * Notional;
  strike = Strike;
END;

IF Floor > 0 THEN
  IF SquaredPayoff == 1 THEN
    floor = pow(Floor, 2);
  ELSE
    floor = Floor;
  END;
  realisedVariation = max(floor * strike, realisedVariation);
END;
IF Cap > 0 THEN
  IF SquaredPayoff == 1 THEN
    cap = pow(Cap, 2);
  ELSE
    cap = Cap;
  END;
  realisedVariation = min(cap * strike, realisedVariation);
END;

payoff = LongShort * knockedIn * (daysBeforeKO / expectedN) *
         currentNotional * (realisedVariation - strike);

Swap = PAY(payoff, ValuationSchedule[SIZE(ValuationSchedule)],
           SettlementDate, PayCcy);
```

Listing 154: Payoff script for a CorridorVarianceSwap.

```
REQUIRE {Notional >= 0} AND {Strike >= 0};
REQUIRE UpperBarrierLevel >= LowerBarrierLevel;

FOR d IN (2, SIZE(ValuationSchedule), 1) DO
  currPrice = Underlying(ValuationSchedule[d]);
  prevPrice = Underlying(ValuationSchedule[d-1]);

  IF {CountBothObservations == 1 AND
      currPrice >= LowerBarrierLevel AND currPrice <= UpperBarrierLevel AND
      prevPrice >= LowerBarrierLevel AND prevPrice <= UpperBarrierLevel} OR
     {CountBothObservations == -1 AND
      prevPrice >= LowerBarrierLevel AND prevPrice <= UpperBarrierLevel} THEN
        realisedVariance = realisedVariance + pow(ln(currPrice/prevPrice), 2);
        accruedDays = accruedDays + 1;
  END;
END;

expectedN = SIZE(ValuationSchedule) - 1;
realisedVariance = (252/expectedN) * realisedVariance;

IF AccrualAdjustment == -1 THEN
  accruedDays = expectedN;
END;

IF SquaredPayoff == 1 THEN
  realisedVariation = realisedVariance;
  currentNotional = pow(100, 2) * Notional / (2 * 100 * Strike);
  strike = pow(Strike, 2);
  adjustedStrike = (accruedDays / expectedN) * strike;
ELSE
  realisedVariation = sqrt(realisedVariance);
  currentNotional = 100 * Notional;
  strike = Strike;
  adjustedStrike = sqrt(accruedDays/expectedN) * strike;
END;

IF Floor > 0 THEN
  IF SquaredPayoff == 1 THEN
    floor = pow(Floor, 2);
  ELSE
    floor = Floor;
  END;
  realisedVariation = max(floor * adjustedStrike, realisedVariation);
END;
IF Cap > 0 THEN
  IF SquaredPayoff == 1 THEN
    cap = pow(Cap, 2);
  ELSE
    cap = Cap;
  END;
  realisedVariation = min(cap * adjustedStrike, realisedVariation);
END;

payoff = LongShort * currentNotional * (realisedVariation - adjustedStrike);

Swap = PAY(payoff, ValuationSchedule[SIZE(ValuationSchedule)],
           SettlementDate, PayCcy);
```

Listing 155: Payoff script for a KIKOCorridorVarianceSwap.

```
REQUIRE {Notional >= 0} AND {Strike >= 0} AND {KIKOBarrierLevel > 0};
REQUIRE CorridorUpperBarrierLevel >= CorridorLowerBarrierLevel;

n = SIZE(ValuationSchedule);

alive = 1;
FOR d IN (2, n, 1) DO
  currPrice = Underlying(ValuationSchedule[d]);
  prevPrice = Underlying(ValuationSchedule[d-1]);

  IF alive == 1 THEN
    IF {CountBothObservations == 1 AND
        currPrice >= CorridorLowerBarrierLevel AND currPrice <= CorridorUpperBarrierLevel AND
        prevPrice >= CorridorLowerBarrierLevel AND prevPrice <= CorridorUpperBarrierLevel} OR
       {CountBothObservations == -1 AND
        prevPrice >= CorridorLowerBarrierLevel AND prevPrice <= CorridorUpperBarrierLevel} THEN
      realisedVariance = realisedVariance + pow(ln(currPrice/prevPrice), 2);
      accruedDays = accruedDays + 1;
    END;

    IF {KIKOBarrierType == 3 AND currPrice <= KIKOBarrierLevel} OR
       {KIKOBarrierType == 4 AND currPrice >= KIKOBarrierLevel} THEN
      alive = 0;
    END;
  END;

  IF knockedIn == 0 THEN
    IF {KIKOBarrierType == 1 AND currPrice <= KIKOBarrierLevel} OR
       {KIKOBarrierType == 2 AND currPrice >= KIKOBarrierLevel} THEN
      knockedIn = 1;
    END;
  END;

  IF {alive == 0 OR d == n} AND {calculated == 0} THEN
    calculated = 1;
    expectedN = n - 1;
    realisedVariance = (252/expectedN) * realisedVariance;

    IF AccrualAdjustment == -1 THEN
      accruedDays = expectedN;
    END;

    currentNotional = pow(100, 2) * Notional / (2 * 100 * Strike);
    strike = pow(Strike, 2);
    adjustedStrike = (accruedDays / expectedN) * strike;

    IF Floor > 0 THEN
      floor = pow(Floor, 2);
      realisedVariance = max(floor * adjustedStrike, realisedVariance);
    END;
    IF Cap > 0 THEN
      cap = pow(Cap, 2);
      realisedVariance = min(cap * adjustedStrike, realisedVariance);
    END;

    IF {{KIKOBarrierType == 1 OR KIKOBarrierType == 2} AND {knockedIn == 1}} OR
       {{KIKOBarrierType == 3 OR KIKOBarrierType == 4} AND {alive == 0}} THEN
      TriggerProbability = 1;
    END;

    IF KIKOBarrierType == 3 OR KIKOBarrierType == 4 THEN
      knockedIn = 1;
    END;

    payoff = LongShort * knockedIn * currentNotional
             * (realisedVariance - adjustedStrike);

    Swap = LOGPAY(payoff, ValuationSchedule[d], SettlementSchedule[d], PayCcy);
  END;
END;
```

Listing 156: Payoff script for a ConditionalVarianceSwap01.

```
REQUIRE {Notional >= 0} AND {Strike > 0};

FOR d IN (2, SIZE(ValuationSchedule), 1) DO
  currPrice = Underlying(ValuationSchedule[d]);
  prevPrice = Underlying(ValuationSchedule[d-1]);

  IF {CountBothObservations == 1 AND {
      {{BarrierType == 1 OR BarrierType == 4} AND
        currPrice <= BarrierLevel AND prevPrice <= BarrierLevel} OR
      {{BarrierType == 2 OR BarrierType == 3} AND
        currPrice >= BarrierLevel AND prevPrice >= BarrierLevel}}}
  OR {CountBothObservations == -1 AND {
      {{BarrierType == 1 OR BarrierType == 4} AND
        prevPrice <= BarrierLevel} OR
      {{BarrierType == 2 OR BarrierType == 3} AND
        prevPrice <= BarrierLevel} }}
  THEN
    realisedVariance = realisedVariance + pow(ln(currPrice/prevPrice), 2);
    accruedDays = accruedDays + 1;
  END;
END;

expectedN = SIZE(ValuationSchedule) - 1;
realisedVariance = (252/expectedN) * realisedVariance;

IF AccrualAdjustment == -1 THEN
  accruedDays = expectedN;
END;

IF SquaredPayoff == 1 THEN
  realisedVariation = realisedVariance;
  currentNotional = pow(100, 2) * Notional / (2 * 100 * Strike);
  strike = pow(Strike, 2);
  adjustedStrike = (accruedDays / expectedN) * strike;
ELSE
  realisedVariation = sqrt(realisedVariance);
  currentNotional = 100 * Notional;
  strike = Strike;
  adjustedStrike = sqrt(accruedDays/expectedN) * strike;
END;

IF Floor > 0 THEN
  IF SquaredPayoff == 1 THEN
    floor = pow(Floor, 2);
  ELSE
    floor = Floor;
  END;
  realisedVariation = max(floor * adjustedStrike, realisedVariation);
END;
IF Cap > 0 THEN
  IF SquaredPayoff == 1 THEN
    cap = pow(Cap, 2);
  ELSE
    cap = Cap;
  END;
  realisedVariation = min(cap * adjustedStrike, realisedVariation);
END;

payoff = LongShort * currentNotional * (realisedVariation - adjustedStrike);

Swap = PAY(payoff, ValuationSchedule[SIZE(ValuationSchedule)],
           SettlementDate, PayCcy);
```

Listing 157: Payoff script for a ConditionalVarianceSwap02.

```
REQUIRE {Notional >= 0} AND {Strike > 0} AND {VarianceReference > 0};

FOR d IN (2, SIZE(ValuationSchedule), 1) DO
  currPrice = Underlying(ValuationSchedule[d]);
  prevPrice = Underlying(ValuationSchedule[d-1]);

  IF {CountBothObservations == 1 AND {
      {{BarrierType == 1 OR BarrierType == 4} AND
        currPrice <= BarrierLevel AND prevPrice <= BarrierLevel} OR
      {{BarrierType == 2 OR BarrierType == 3} AND
        currPrice >= BarrierLevel AND prevPrice >= BarrierLevel}}}
  OR {CountBothObservations == -1 AND {
      {{BarrierType == 1 OR BarrierType == 4} AND
        prevPrice <= BarrierLevel} OR
      {{BarrierType == 2 OR BarrierType == 3} AND
        prevPrice <= BarrierLevel} }}
  THEN
    realisedVariance = realisedVariance + pow(ln(currPrice/prevPrice), 2);
    accruedDays = accruedDays + 1;
  END;
END;

expectedN = SIZE(ValuationSchedule) - 1;
realisedVariance = (252/expectedN) * realisedVariance;

IF AccrualAdjustment == -1 THEN
  accruedDays = expectedN;
END;

IF SquaredPayoff == 1 THEN
  realisedVariation = realisedVariance;
  currentNotional = pow(100, 2) * Notional / (2 * 100 * VarianceReference);
  strike = pow(Strike, 2);
  adjustedStrike = (accruedDays / expectedN) * strike;
ELSE
  realisedVariation = sqrt(realisedVariance);
  currentNotional = 100 * Notional;
  strike = Strike;
  adjustedStrike = sqrt(accruedDays/expectedN) * strike;
END;

IF Floor > 0 THEN
  IF SquaredPayoff == 1 THEN
    floor = pow(Floor, 2);
  ELSE
    floor = Floor;
  END;
  realisedVariation = max(floor * adjustedStrike, realisedVariation);
END;
IF Cap > 0 THEN
  IF SquaredPayoff == 1 THEN
    cap = pow(Cap, 2);
  ELSE
    cap = Cap;
  END;
  realisedVariation = min(cap * adjustedStrike, realisedVariation);
END;

payoff = LongShort * currentNotional * (realisedVariation - adjustedStrike);

Swap = PAY(payoff, ValuationSchedule[SIZE(ValuationSchedule)],
           SettlementDate, PayCcy);
```

Listing 159: Payoff script for a PairwiseVarianceSwap.

```
REQUIRE {SIZE(Underlyings) == 2} AND {SIZE(UnderlyingStrikes) == 2}
REQUIRE {SIZE(UnderlyingNotionals) == 2} AND {UnderlyingStrikes[1] >= 0};
REQUIRE {UnderlyingStrikes[2] >= 0} AND {BasketStrike >= 0};
REQUIRE {UnderlyingNotionals[1] >= 0} AND {UnderlyingNotionals[2] >= 0};
REQUIRE {BasketNotional >= 0} AND {PayoffLimit > 0};
REQUIRE {SIZE(ValuationSchedule) == SIZE(LaggedValuationSchedule)};

FOR d IN (1, SIZE(ValuationSchedule)-1, 1) DO
  performance1 = ln(Underlyings[1](LaggedValuationSchedule[d]) /
                    Underlyings[1](ValuationSchedule[d]));
  performance2 = ln(Underlyings[2](LaggedValuationSchedule[d]) /
                    Underlyings[2](ValuationSchedule[d]));
  basketPerformance = (performance1 + performance2) / 2;

  realisedVariance1 = realisedVariance1 + pow(performance1, 2);
  realisedVariance2 = realisedVariance2 + pow(performance2, 2);
  realisedVarianceBasket = realisedVarianceBasket + pow(basketPerformance, 2);
END;

expectedN = SIZE(ValuationSchedule) - 1;
realisedVariance1 = 252 / (expectedN * AccrualLag) * realisedVariance1;
realisedVariance2 = 252 / (expectedN * AccrualLag) * realisedVariance2;
realisedVarianceBasket = 252 / (expectedN * AccrualLag) * realisedVarianceBasket;

currentNotional1 = pow(100, 2) * UnderlyingNotionals[1] /
                                  (2 * 100 * UnderlyingStrikes[1]);
currentNotional2 = pow(100, 2) * UnderlyingNotionals[2] /
                                  (2 * 100 * UnderlyingStrikes[2]);
currentNotionalBasket = pow(100, 2) * BasketNotional / (2 * 100 * BasketStrike);
strike1 = pow(UnderlyingStrikes[1], 2);
strike2 = pow(UnderlyingStrikes[2], 2);
strikeBasket = pow(BasketStrike, 2);

IF Floor > 0 THEN
  floor = pow(Floor, 2);
  realisedVariance1 = max(floor * strike1, realisedVariance1);
  realisedVariance2 = max(floor * strike2, realisedVariance2);
  realisedVarianceBasket = max(floor * strikeBasket, realisedVarianceBasket);
END;
IF Cap > 0 THEN
  cap = pow(Cap, 2);
  realisedVariance1 = min(cap * strike1, realisedVariance1);
  realisedVariance2 = min(cap * strike2, realisedVariance2);
  realisedVarianceBasket = min(cap * strikeBasket, realisedVarianceBasket);
END;

equityAmount1 = currentNotional1 * (realisedVariance1 - strike1);
equityAmount2 = currentNotional2 * (realisedVariance2 - strike2);
equityAmountBasket = currentNotionalBasket * (realisedVarianceBasket - strikeBasket);
pairEquityAmount = equityAmount1 + equityAmount2 + equityAmountBasket;

maxPairEquityAmount = PayoffLimit * (abs(UnderlyingNotionals[1]) +
                                     abs(UnderlyingNotionals[2]));
minPairEquityAmount = -maxPairEquityAmount;

pairEquityAmount = max(minPairEquityAmount, pairEquityAmount);
pairEquityAmount = min(maxPairEquityAmount, pairEquityAmount);

Swap = PAY(LongShort * pairEquityAmount, ValuationSchedule[SIZE(ValuationSchedule)],
           SettlementDate, PayCcy);
```

Listing 160: Payoff script for a VarianceDispersionSwap.

```
D = SIZE(ValuationSchedule);
expectedN = D - 1;

FOR u IN (1, n1, 1) DO
  FOR d IN (2, D, 1) DO
    currPrice = Underlyings1[u](ValuationSchedule[d]);
    prevPrice = Underlyings1[u](ValuationSchedule[d-1]);
    realisedVariance = realisedVariance + pow(ln(currPrice/prevPrice), 2);
  END;

  realisedVariance = (252/expectedN) * realisedVariance + Spreads1[u];
  currentNotional = pow(100, 2) * Notionals1[u] / (2 * 100 * Strikes1[u]);
  strike = pow(Strikes1[u], 2);

  IF Floors1[u] > 0 THEN
    floor = pow(Floors1[u], 2);
    realisedVariance = max(floor * strike, realisedVariance);
  END;
  IF Caps1[u] > 0 THEN
    cap = pow(Caps1[u], 2);
    realisedVariance = min(cap * strike, realisedVariance);
  END;
  payoff1 = payoff1 + currentNotional * (realisedVariance - strike) * Weights1[u];
END;

FOR u IN (1, n2, 1) DO
  FOR d IN (2, D, 1) DO
    currPrice = Underlyings2[u](ValuationSchedule[d]);
    prevPrice = Underlyings2[u](ValuationSchedule[d-1]);
    realisedVariance = realisedVariance + pow(ln(currPrice/prevPrice), 2);
  END;

  realisedVariance = (252/expectedN) * realisedVariance + Spreads2[u];
  currentNotional = pow(100, 2) * Notionals2[u] / (2 * 100 * Strikes2[u]);
  strike = pow(Strikes2[u], 2);

  IF Floors2[u] > 0 THEN
    floor = pow(Floors2[u], 2);
    realisedVariance = max(floor * strike, realisedVariance);
  END;
  IF Caps2[u] > 0 THEN
    cap = pow(Caps2[u], 2);
    realisedVariance = min(cap * strike, realisedVariance);
  END;
  payoff2 = payoff2 + currentNotional * (realisedVariance - strike) * Weights2[u];
END;

payoff = LongShort * (payoff1 - payoff2);

Swap = PAY(payoff, ValuationSchedule[D], SettlementDate, PayCcy);

NUMBER currentNotional1, currentNotional2;

FOR u IN (1, n1, 1) DO
  currentNotional1 = currentNotional1 + (pow(100, 2) * Notionals1[u]
                                         / (2 * 100 * Strikes1[u]));
END;
FOR u IN (1, n2, 1) DO
  currentNotional2 = currentNotional2 + (pow(100, 2) * Notionals2[u]
                                         / (2 * 100 * Strikes2[u]));
END;
```

Listing 161: Payoff script for a CorridorVarianceDispersionSwap.

```
N = SIZE(Underlyings1);
D = SIZE(ValuationSchedule);
expectedN = D - 1;

FOR u IN (1, N, 1) DO
  FOR d IN (2, D, 1) DO
    currPrice1 = Underlyings1[u](ValuationSchedule[d]);
    prevPrice1 = Underlyings1[u](ValuationSchedule[d-1]);

    IF {CountBothObservations == 1 AND
        currPrice1 >= LowerBarrierLevels[u] AND currPrice1 <= UpperBarrierLevels[u] AND
        prevPrice1 >= LowerBarrierLevels[u] AND prevPrice1 <= UpperBarrierLevels[u]} OR
       {CountBothObservations == -1 AND
        prevPrice >= LowerBarrierLevels[u] AND prevPrice <= UpperBarrierLevels[u]} THEN
          currPrice2 = Underlyings2[u](ValuationSchedule[d]);
          prevPrice2 = Underlyings2[u](ValuationSchedule[d-1]);
          realisedVariance1 = realisedVariance1 + pow(ln(currPrice1/prevPrice1), 2);
          realisedVariance2 = realisedVariance2 + pow(ln(currPrice2/prevPrice2), 2);
          accruedDays = accruedDays + 1;
    END;
  END;

  IF AccrualAdjustment == -1 THEN
    accruedDays = expectedN;
  END;

  realisedVariance1 = (252/expectedN) * realisedVariance1 + Spreads1[u];
  realisedVariance2 = (252/expectedN) * realisedVariance2 + Spreads2[u];

  currentNotional1 = pow(100, 2) * Notionals1[u] / (2 * 100 * Strikes1[u]);
  currentNotional2 = pow(100, 2) * Notionals2[u] / (2 * 100 * Strikes2[u]);

  strike1 = pow(Strikes1[u], 2);
  strike2 = pow(Strikes2[u], 2);

  adjustedStrike1 = (accruedDays / expectedN) * strike1;
  adjustedStrike2 = (accruedDays / expectedN) * strike2;

  IF Floors1[u] > 0 THEN
    floor1 = pow(Floors1[u], 2);
    realisedVariance1 = max(floor1 * adjustedStrike1, realisedVariance1);
  END;
  IF Floors2[u] > 0 THEN
    floor2 = pow(Floors2[u], 2);
    realisedVariance2 = max(floor2 * adjustedStrike2, realisedVariance2);
  END;
  IF Caps1[u] > 0 THEN
    cap1 = pow(Caps1[u], 2);
    realisedVariance1 = max(cap1 * adjustedStrike1, realisedVariance1);
  END;
  IF Caps2[u] > 0 THEN
    cap2 = pow(Caps2[u], 2);
    realisedVariance2 = max(cap2 * adjustedStrike2, realisedVariance2);
  END;

  payoff1 = currentNotional1 * (realisedVariance1 - adjustedStrike1);
  payoff2 = currentNotional2 * (realisedVariance2 - adjustedStrike2);
  payoff = payoff + LongShort * Weights[u] * (payoff1 - payoff2);
END;

Swap = PAY(payoff, ValuationSchedule[D], SettlementDate, PayCcy);
```

Listing 162: Payoff script for a KOCorridorVarianceDispersionSwap.

```
N = SIZE(Underlyings1);
Dk = SIZE(KnockOutSchedule);
expectedN = D - 1;

FOR d IN (1, Dk, 1) DO
  IF KnockOutSchedule[d] > VarianceAccrualStartDate THEN
    expectedN = expectedN + 1;
  END;
END;

alive = 1;
FOR d IN (1, Dk, 1) DO
  IF alive == 1 THEN
    IF KnockOutSchedule[d] > VarianceAccrualStartDate THEN
      FOR u IN (1, N, 1) DO
        currPrice1 = Underlyings1[u](KnockOutSchedule[d]);
        prevPrice1 = Underlyings1[u](KnockOutSchedule[d-1]);

        IF {CountBothObservations == 1 AND
            currPrice1 >= CorridorLowerBarrierLevels[u] AND currPrice1 <= CorridorUpperBarrierLevels[u] AND
            prevPrice1 >= CorridorLowerBarrierLevels[u] AND prevPrice1 <= CorridorUpperBarrierLevels[u]} OR
           {CountBothObservations == -1 AND
            prevPrice >= CorridorLowerBarrierLevels[u] AND prevPrice <= CorridorUpperBarrierLevels[u]} THEN
          currPrice2 = Underlyings2[u](KnockOutSchedule[d]);
          prevPrice2 = Underlyings2[u](KnockOutSchedule[d-1]);
          realisedVariance1 = realisedVariance1 + pow(ln(currPrice1/prevPrice1), 2);
          realisedVariance2 = realisedVariance2 + pow(ln(currPrice2/prevPrice2), 2);
          accruedDays = accruedDays + 1;
        END;
      END;
    END;

    FOR u IN (1, N, 1) DO
      currPrice1 = Underlyings1[u](KnockOutSchedule[d]);
      IF currPrice1 <= KOLowerBarrierLevels[u] OR currPrice1 >= KOUpperBarrierLevels[u] THEN
        alive = 0;
      END;
    END;
  END;

  IF {alive == 0 OR d == Dk} AND {calculated == 0} THEN
    calculated = 1;

    realisedVariance1 = (252/expectedN) * realisedVariance1 + Spreads1[u];
    realisedVariance2 = (252/expectedN) * realisedVariance2 + Spreads2[u];

    IF AccrualAdjustment == -1 THEN
      accruedDays = expectedN;
    END;

    currentNotional1 = pow(100, 2) * Notionals1[u] / (2 * 100 * Strikes1[u]);
    currentNotional2 = pow(100, 2) * Notionals2[u] / (2 * 100 * Strikes2[u]);

    strike1 = pow(Strikes1[u], 2);
    strike2 = pow(Strikes2[u], 2);

    adjustedStrike1 = (accruedDays / expectedN) * strike1;
    adjustedStrike2 = (accruedDays / expectedN) * strike2;

    IF Floors1[u] > 0 THEN
      floor1 = pow(Floors1[u], 2);
      realisedVariance1 = max(floor1 * adjustedStrike1, realisedVariance1);
    END;
    IF Floors2[u] > 0 THEN
      floor2 = pow(Floors2[u], 2);
      realisedVariance2 = max(floor2 * adjustedStrike2, realisedVariance2);
    END;
    IF Caps1[u] > 0 THEN
      cap1 = pow(Caps1[u], 2);
      realisedVariance1 = max(cap1 * adjustedStrike1, realisedVariance1);
    END;
    IF Caps2[u] > 0 THEN
      cap2 = pow(Caps2[u], 2);
      realisedVariance2 = max(cap2 * adjustedStrike2, realisedVariance2);
    END;

    payoff1 = currentNotional1 * (realisedVariance1 - adjustedStrike1);
    payoff2 = currentNotional2 * (realisedVariance2 - adjustedStrike2);
    payoff = payoff + LongShort * Weights[u] * (payoff1 - payoff2);
```

Listing 163: Payoff script for a GammaSwap.

```
REQUIRE {Notional >= 0} AND {Strike >= 0};

NUMBER d, expectedN, realisedVariance, currPrice, prevPrice, presPrice, currentNotional;
NUMBER payoff, realisedVariation, strike;

presPrice = Underlying(ValuationSchedule[1]);

FOR d IN (2, SIZE(ValuationSchedule), 1) DO
  currPrice = Underlying(ValuationSchedule[d]);
  prevPrice = Underlying(ValuationSchedule[d-1]);
  realisedVariance = realisedVariance + (currPrice/presPrice) * pow(ln(currPrice/prevPrice), 2);
END;

expectedN = SIZE(ValuationSchedule) - 1;
realisedVariance = (252/expectedN) * realisedVariance;

realisedVariation = realisedVariance;
currentNotional = pow(100, 2) * Notional / (2 * 100 * Strike);
strike = pow(Strike, 2);

payoff = LongShort * currentNotional * (realisedVariation -
            strike);

Swap = PAY(payoff, ValuationSchedule[SIZE(ValuationSchedule)],
           SettlementDate, PayCcy);
```

Listing 165: Payoff script for a BasketVarianceSwap.

```
NUMBER sumOfWeights;
FOR i IN (1, n, 1) DO
  sumOfWeights = sumOfWeights + Weights[i];
END;
REQUIRE sumOfWeights == 1;

NUMBER d, expectedN, currPrice[n], prevPrice[n];
NUMBER realisedVariance, basketVariation, realisedVariation;
NUMBER strike, cap, floor, currentNotional, payoff;

FOR d IN (2, SIZE(ValuationSchedule), 1) DO
  basketVariation = 0;
  FOR i IN (1, n, 1) DO
    currPrice[i] = Underlyings[i](ValuationSchedule[d]);
    prevPrice[i] = Underlyings[i](ValuationSchedule[d-1]);
    basketVariation = basketVariation + Weights[i] * ln(currPrice[i]/prevPrice[i]);
  END;
  realisedVariance = realisedVariance + pow(basketVariation, 2);
END;

expectedN = SIZE(ValuationSchedule) - 1;
realisedVariance = (252/expectedN) * realisedVariance;

IF SquaredPayoff == 1 THEN
  realisedVariation = realisedVariance;
  currentNotional = pow(100, 2) * Notional / (2 * 100 * Strike);
  strike = pow(Strike, 2);
ELSE
  realisedVariation = sqrt(realisedVariance);
  currentNotional = 100 * Notional;
  strike = Strike;
END;

IF Floor > 0 THEN
  IF SquaredPayoff == 1 THEN
    floor = pow(Floor, 2);
  ELSE
    floor = Floor;
  END;
  realisedVariation = max(floor * strike, realisedVariation);
END;
IF Cap > 0 THEN
  IF SquaredPayoff == 1 THEN
    cap = pow(Cap, 2);
  ELSE
    cap = Cap;
  END;
  realisedVariation = min(cap * strike, realisedVariation);
END;

payoff = LongShort * currentNotional * (realisedVariation - strike);

Swap = PAY(payoff, ValuationSchedule[SIZE(ValuationSchedule)],
           SettlementDate, PayCcy);
```

Listing 203: Commodity fixed leg outline.

```
<LegData>
  <LegType>CommodityFixed</LegType>
  <Payer>...</Payer>
  <Currency>...</Currency>
  <PaymentConvention>...</PaymentConvention>
  <PaymentLag>...</PaymentLag>
  <PaymentCalendar>...</PaymentCalendar>
  <ScheduleData>
    ...
  </ScheduleData>
  <PaymentDates>
    <PaymentDate>...</PaymentDate>
  </PaymentDates>
  <CommodityFixedLegData>
    ...
  </CommodityFixedLegData>
</LegData>
```

Listing 204: Commodity fixed leg data outline.

```
<CommodityFixedLegData>
  <Quantities>
    <Quantity>...</Quantity>
  </Quantities>
  <Prices>
    <Price>...</Price>
  </Prices>
  <CommodityPayRelativeTo>...</CommodityPayRelativeTo>
  <Tag>...</Tag>
</CommodityFixedLegData>
```

Listing 205: Commodity floating leg outline.

```
<LegData>
  <LegType>CommodityFloating</LegType>
  <Payer>...</Payer>
  <Currency>...</Currency>
  <PaymentConvention>...</PaymentConvention>
  <PaymentLag>...</PaymentLag>
  <PaymentCalendar>...</PaymentCalendar>
  <ScheduleData>
    ...
  </ScheduleData>
  <PaymentDates>
    <PaymentDate>...</PaymentDate>
  </PaymentDates>
  <CommodityFloatingLegData>
    ...
  </CommodityFloatingLegData>
</LegData>
```

Listing 206: `ScheduleData` *node for monthly periods.*

```
<ScheduleData>
  <Rules>
    <StartDate>2020-01-01</StartDate>
    <EndDate>2020-04-30</EndDate>
    <Tenor>1M</Tenor>
    <Calendar>NullCalendar</Calendar>
    <Convention>Unadjusted</Convention>
    <TermConvention>Unadjusted</TermConvention>
    <Rule>Backward</Rule>
    <EndOfMonth>true</EndOfMonth>
    <AdjustEndDateToPreviousMonthEnd>false</AdjustEndDateToPreviousMonthEnd>
  </Rules>
</ScheduleData>
```

Listing 207: `ScheduleData` *node for explicit periods.*

```
<ScheduleData>
  <Dates>
    <Calendar>NullCalendar</Calendar>
    <Convention>Unadjusted</Convention>
    <Dates>
      <Date>2019-11-21</Date>
      <Date>2019-12-19</Date>
      <Date>2020-01-21</Date>
      <Date>2020-02-20</Date>
      <Date>2020-03-20</Date>
    </Dates>
  </Dates>
</ScheduleData>
```

Listing 208: Commodity floating leg data outline.

```
<CommodityFloatingLegData>
  <Name>...</Name>
  <PriceType>...</PriceType>
  <Quantities>
    <Quantity>...</Quantity>
  </Quantities>
  <CommodityQuantityFrequency>...</CommodityQuantityFrequency>
  <CommodityPayRelativeTo>...</CommodityPayRelativeTo>
  <Spreads>
    <Spread>...</Spread>
  </Spreads>
  <Gearings>
    <Gearing>...</Gearing>
  </Gearings>
  <PricingDateRule>...</PricingDateRule>
  <PricingCalendar>...</PricingCalendar>
  <PricingLag>...</PricingLag>
  <PricingDates>
    <PricingDate>...</PricingDate>
  </PricingDates>
  <IsAveraged>...</IsAveraged>
  <IsInArrears>...</IsInArrears>
  <FutureMonthOffset>...</FutureMonthOffset>
  <DeliveryRollDays>...</DeliveryRollDays>
  <IncludePeriodEnd>...</IncludePeriodEnd>
  <ExcludePeriodStart>...</ExcludePeriodStart>
  <HoursPerDay>...</HoursPerDay>
  <UseBusinessDays>...</UseBusinessDays>
  <Tag>...</Tag>
  <DailyExpiryOffset>...</DailyExpiryOffset>
</CommodityFloatingLegData>
```

Listing 209: Example WTI averaging floating leg, first approach.

```
<LegData>
  <LegType>CommodityFloating</LegType>
  <Payer>true</Payer>
  <Currency>USD</Currency>
  <PaymentLag>2</PaymentLag>
  <PaymentConvention>Following</PaymentConvention>
  <PaymentCalendar>US-NYSE</PaymentCalendar>
  <CommodityFloatingLegData>
    <Name>NYMEX:CL</Name>
    <PriceType>FutureSettlement</PriceType>
    <Quantities>
      <Quantity>5000</Quantity>
    </Quantities>
    <IsAveraged>true</IsAveraged>
    <FutureMonthOffset>0</FutureMonthOffset>
  </CommodityFloatingLegData>
  <ScheduleData>
    <Rules>
      <StartDate>2019-09-01</StartDate>
      <EndDate>2019-10-31</EndDate>
      <Tenor>1M</Tenor>
      <Calendar>NullCalendar</Calendar>
      <Convention>Unadjusted</Convention>
      <TermConvention>Unadjusted</TermConvention>
      <Rule>Backward</Rule>
      <EndOfMonth>true</EndOfMonth>
    </Rules>
  </ScheduleData>
</LegData>
```

Listing 210: Example WTI averaging floating leg, second approach.

```
<CommodityFloatingLegData>
<Name>NYMEX:CSX</Name>
<PriceType>FutureSettlement</PriceType>
<Quantities>
  <Quantity>5000</Quantity>
</Quantities>
<IsAveraged>true</IsAveraged>
<FutureMonthOffset>0</FutureMonthOffset>
</CommodityFloatingLegData>
```

Listing 211: Equity Margin leg outline.

```
<LegData>
  <LegType>EquityMargin</LegType>
  <Payer>true</Payer>
  <Currency>EUR</Currency>
  <PaymentConvention>Following</PaymentConvention>
  <PaymentLag>2D</PaymentLag>
  <PaymentCalendar>TARGET</PaymentCalendar>
  <ScheduleData>
       <Dates>
        <Dates>
          <Date>2019-12-31</Date>
          <Date>2020-03-30</Date>
          <Date>2020-06-30</Date>
          <Date>2020-09-30</Date>
          <Date>2020-12-30</Date>
          <Date>2021-03-30</Date>
        </Dates>
     </Dates>
  </ScheduleData>
  <PaymentDates>
    <PaymentDate>...</PaymentDate>
  </PaymentDates>
  <EquityMarginLegData>
    ...
  </EquityMarginLegData>
</LegData>
```

Listing 212: Equity margin leg data outline.

```
<EquityMarginLegData>
  <Rates>
    <Rate>0.003</Rate>
  </Rates>
  <InitialMarginFactor>0.12</InitialMarginFactor>
  <Multiplier>10</Multiplier>
  <EquityLegData>
     <ReturnType>Total</ReturnType>
     <Name>RIC:.STOXX50E</Name>
     <InitialPrice>2946</InitialPrice>
     <NotionalReset>false</NotionalReset>
     <FixingDays>2</FixingDays>
  </EquityLegData>
</EquityMarginLegData>
```

Listing 232: FormulaBasedLegData configuration.

```
<LegData>
  <LegType>FormulaBased</LegType>
  <Payer>true</Payer>
  <Currency>EUR</Currency>
  <PaymentConvention>MF</PaymentConvention>
  <PaymentLag>2</PaymentLag>
  <PaymentCalendar>TARGET</PaymentCalendar>
  <DayCounter>A360</DayCounter>
   ...
  <FormulaBasedLegData>
    <Index>gtZero({USD-CMS-5Y}-0.03)*
              max(min(9.0*({EUR-CMS-10Y}-{GBP-CMS-2Y})+0.02,0.08),0.0)</Index>
    <IsInArrears>false</IsInArrears>
    <FixingDays>2</FixingDays>
  </FormulaBasedLegData>
   ...
</LegData>
```

2.4 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 13: 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 Modif* *NEAREST* (takes future date in case of equal distance)

Table 14: 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 15: 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 16: 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 17: 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 (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 18: 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 20: 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 21: 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 22: 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 23: 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 24: 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 25: 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 26: 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 27: 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 28: Allowable Values for Exchange

Boolean nodes	
Node Value	**Evaluates To**
Y, YES, TRUE, true, 1	true
N, NO, FALSE, false, 0	false

Table 29: Allowable values for boolean node

3 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.

3.1 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 233: 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.

3.2 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 234: 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 15.

- `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 19.

- `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 15.
- `ApplyInitialMargin`: Apply (dynamic) initial Margin in addition to variation margin
  Allowable values: Boolean node, the set of allowable values is given in Table 29.
- `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 29.
- `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 29.

4 Scripted Trade

The scripted trade provides a generic interface for representing exotic products.

The trade's payoff is defined in terms of a *payoff script* using a *scripting langage* described in appendix 4.5. The payoff script can depend on single variables or sets of variables in a single risk class or across several risk classes. The payoff script currently supports EQ, FX (including Precious Metals), Commodity, Inflation and IR variables, i.e. Equity spot prices, FX spot rates, Commodity spot or future prices, IBOR index or CMS index fixings.

4.1 General Structure

A scripted trade comprises

- external data that parametrises the payoff script
- the payoff script

In the following example, a plain vanilla equity call or put option is described by providing the expiry and settlement date, the strike, a put / call indicator and the underlying index as external data in the `Data` node, and the payoff script is referenced by the `ScriptName` node:

```
<Trade id="VanillaOption">
  <TradeType>ScriptedTrade</TradeType>
  <Envelope/>
  <ScriptedTradeData>
    <ScriptName>EuropeanOption</ScriptName>
    <Data>
      <Event>
        <Name>Expiry</Name>
        <Value>2020-02-09</Value>
      </Event>
      <Event>
        <Name>Settlement</Name>
        <Value>2020-02-15</Value>
      </Event>
      <Number>
        <Name>Strike</Name>
        <Value>1200</Value>
      </Number>
      <Number>
        <Name>PutCall</Name>
        <Value>1</Value>
      </Number>
      <Index>
        <Name>Underlying</Name>
        <Value>EQ-RIC:.SPX</Value>
      </Index>
      <Currency>
        <Name>PayCcy</Name>
        <Value>USD</Value>
      </Currency>
    </Data>
  </ScriptedTradeData>
</Trade>
```

The script itself is defined in a script library, which is part of the static data setup of

the application like bond or credit index static data, as follows

```
<ScriptLibrary>
  <Script>
    <Name>EuropeanOption</Name>
    <ProductTag>SingleAssetOption({AssetClass})</ProductTag>
    <Script>
      <Code><![CDATA[
          Option = PAY(max( PutCall * (Underlying(Expiry) - Strike), 0 ),
                                                  Expiry, Settlement, PayCcy);
        ]]></Code>
      <NPV>Option</NPV>
      <Results>
        <Result>ExerciseProbability</Result>
      </Results>
    </Script>
  </Script>
  <Script>
  ...
  </Script>
  ...
</ScriptLibrary>
```

Many scripted trades can thus share the same script, and the payoff scripts can be managed and maintained centrally. Each scipt defines an exotic product type. And adding a product type to the library is a configuration rather than a code release change.

4.2 Data Types

The `Data` node contains definitions for external variables that can be used in the script. These variables can either be scalars or one-dimensional arrays with fixed size. The supported data types are

- Event: a date
- Number: a real number
- Counter: an integer
- Index: an EQ, FX or COMM index
- Currency: a currency
- Daycounter: a day count convention

4.2.1 Event

An event is defined either as a scalar

```
    <Event>
      <Name>Expiry</Name>
      <Value>2020-02-09</Value>
    </Event>
```

or as an array of dates using ORE's ScheduleData node, e.g.

```
<Event>
  <Name>ExerciseDates</Name>
  <ScheduleData>
    <Rules>
     <StartDate>2016-02-06</StartDate>
     <EndDate>2016-05-06</EndDate>
     <Tenor>1D</Tenor>
     <Calendar>TARGET,US</Calendar>
     <Convention>F</Convention>
     <Rule>Forward</Rule>
   </Rules>
  </ScheduleData>
</Event>
```

using a rule based schedule or

```
<Event>
  <Name>ValuationDates</Name>
  <ScheduleData>
    <Dates>
      <Dates>
        <Date>2018-03-10</Date>
        <Date>2019-03-10</Date>
        <Date>2020-03-10</Date>
        <Date>2021-03-10</Date>
        <Date>2022-03-10</Date>
        <Date>2023-03-10</Date>
        <Date>2024-03-11</Date>
      </Dates>
    </Dates>
  </ScheduleData>
</Event>
```

using a list of dates. An array of dates can also be deduced from another, previously defined array by specifying a shift rule which is useful e.g. to generate fixing or payment schedules from accrual schedules, or notification and settlement schedules from exercise date schedules. Example:

```
<Event>
  <Name>FixingDates</Name>
  <DerivedSchedule>
    <BaseSchedule>AccrualDates</BaseSchedule>
    <Shift>-2D</Shift>
    <Calendar>TARGET</Calendar>
    <Convention>MF</Convention>
  </DerivedSchedule>
</Event>
```

4.2.2 Number

A scalar number is defined as

```
<Number>
  <Name>Strike</Name>
  <Value>2147.56</Value>
</Number>
```

and likewise an array of numbers as

```
<Number>
  <Name>EquityNotionalAmount</Name>
  <Values>
    <Value>100000000</Value>
    <Value>90000000</Value>
    <Value>80000000</Value>
  </Values>
</Number>
```

4.2.3 Index

Indices are defined as

```
<Index>
  <Name>Underlying</Name>
  <Value>EQ-RIC:.SPX</Value>
</Index>
```

in case of a scalar and

```
<Index>
  <Name>Underlyings</Name>
  <Values>
    <Value>EQ-UND1</Value>
    <Value>EQ-UND2</Value>
    <Value>EQ-UND3</Value>
    <Value>EQ-UND4</Value>
    <Value>EQ-UND5</Value>
  </Values>
</Index>
```

in case of a vector. Currently, Equity, FX, Commodity, Interest Rate and Inflation indices as well as generic indices are supported, the naming convention follows the standard ORE conventions, i.e.

- Equity: These are declared based on identifier type
  - `EQ-ISIN:{Name}:{Currency}:{Exchange}` for ISIN equities, e.g. `EQ-ISIN:NL0000852580:EUR:XAMS`,
  - `EQ-RIC:{Name}` for RIC equities, e.g. `EQ-RIC:.SPX`,
  - `EQ-FIGI:{Name}` for FIGI equities, e.g. `EQ-FIGI:BBG000BLNNV0`,
  - `EQ-BBG:{Name}` for BBG equities, e.g. `EQ-BBG:BARC LN Equity`.

- FX: `FX-SOURCE-CCY1-CCY2` with foreign currency `CCY1` and domestic currency `CCY2`, as in Table 21. The order of the currencies here is important as they will have different meanings. For a natural payoff, `CCY2` must correspond to the instrument's payment currency. Otherwise, we have a quanto payoff. See section 4.4 for more information.
- Commodity: These are declared as `COMM-{Name}`, with possible extensions as follows (see below also for more information on the meanings of the extensions to the commodity indices):
  - `COMM-NYMEX:CL` (spot),
  - `COMM-NYMEX:CL-2020-09` (future with expiry 01 Sep 2020),
  - `COMM-NYMEX:CL-2020-09-15` (future with expiry 15 Sep 2020).
- Interest Rate: These are declared as `CCY-INDEX-TENOR`, e.g. `EUR-EURIBOR-6M`, `EUR-CMS-10Y`. See Table 19 for a list of allowable IR indices.
- Inflation: These are declared as `{Name}`, with possible extensions as follows(see below also for more information on the meanings of the extensions to the inflation indices):
  - `EUHICPXT` (constant/non-interpolated),
  - `EUHICPXT#F` (flat interpolation),
  - `EUHICPXT#L` (linear interpolation).
- Generic: These are declared as `GENERIC-{Name}`.

Notice that generic indices only provide historical fixings.

For additional information on the equity name conventions see 23 and 2.3.29. In general the equity name can be represented using the string representation IdentifierType:Name:Currency:Exchange for the underlying.
For additional information on the fx index format see 21.
For additional information on the commodity names see 25.

Within the scripting framework there are additional ways to reference commodity indices in a more dynamical fashion: When evaluating a commodity index using the index evaluation operator (see 4.5) as in

```
CommodityUnderlying(ObservationDate)
```

for

- a variable of the form `COMM-{Name}#N`, the index will be resolved to the $N+1$th future with expiry greater than ObservationDate for the given commodity underlying, $N \geq 0$. The parameter N corresponds to the field `FutureMonthOffset` commonly used in commodity trade xml schemas
- the above form can take one or two additional parameters `COMM-{Name}#N#D` or `COMM-{Name}#N#D#{Cal}` where D corresponds to `DeliveryRollDays` and `Cal` is the calendar used to roll the observation date forward before the next future

expiry is looked up. If `Cal` is not given, it defaults to the null calendar (no holidays).

- a variable of the form `COMM-{Name}!N`, the index will be resolved to the Nth future relative to the month and year of the ObservationDate.

In general, if a commodity future is referenced, the observation date should be be less or equal to the expiry date of the future, since no historical fixing will in general be available after the future expiry date. In the future-looking simulation the observation of an expired future is possible, in this case the (model) value of the future is kept constant at its value at the expiry date for observation dates after the expiry date.

When evaluating an inflation index as in

```
CPIIndex(FixingDate)
```

the variable `CPIIndex` can be given in the following ways:

- a variable of the form `EUHICPXT` will evaluate the standard ORE inflation index, which is non-interpolated. This means the result will be constant for all FixingDates in an inflation period (i.e. usually constant for a calendar month)
- a variable of the form `EUHICPXT#F`, i.e. an ORE inflation index name followed by `#F` indicating a flat interpolation of the index, this form is equivalent to the previous form making the interpolation mode explicit as "flat"
- a variable of the form `EUHICPXT#L` where the suffix `#L` indicates linear interpolation of the index, i.e. the result will be the interpolated fixing between the usually monthly inflation index fixings

The use of the interpolation extensions is deprecated. Interpolation of fixings should be handled in the payoff script. The CPI indices are forecasting a flat fixing for given the inflation period. The use of interpolation is still supported but will be eventually removed in a later release.

4.2.4 Currency

A scalar currency variable is defined as

```
<Currency>
  <Name>PayCcy</Name>
  <Value>USD</Value>
</Currency>
```

and an array of currencies as

```
<Currency>
  <Name>BasketCurrencies</Name>
  <Values>
    <Value>USD</Value>
    <Value>GBP</Value>
    <Value>JPY</Value>
```

```
      </Value>
    </Currency>
```

4.2.5 Daycounter

A scalar day count convention variable is defined as

```
    <Daycounter>
      <Name>AccrualDayCounter</Name>
      <Value>Actual/360</Value>
    </Daycounter>
```

and an array of day counters as

```
    <Daycounter>
      <Name>LegAccrualDayCounters</Name>
      <Values>
        <Value>30/360</Value>
        <Value>Actual/360</Value>
      </Value>
    </Currency>
```

4.3 Compact Trade XML

In addition to the trade xml described in 4.1, we support an alternative, more compact, format where the variable names are derived from the node names and the type is given by an attribute. The script is referenced via a name derived from the root node of the trade data. Consider the following example of a one touch option in the original format:

```
<Trade id="SCRIPTED_FX_ONE-TOUCH_OPTION">
  <TradeType>ScriptedTrade</TradeType>
  <Envelope>
    <CounterParty>CPTY_A</CounterParty>
    <NettingSetId>CPTY_A</NettingSetId>
    <AdditionalFields/>
  </Envelope>
  <ScriptedTradeData>
    <ScriptName>OneTouchOption</ScriptName>
    <Data>
      <Event>
        <Name>Settlement</Name>
        <Value>2020-08-01</Value>
      </Event>
      <Event>
        <Name>ObservationDates</Name>
        <ScheduleData>
          <Rules>
            <StartDate>2018-12-28</StartDate>
            <EndDate>2020-08-01</EndDate>
            <Tenor>1D</Tenor>
            <Calendar>US</Calendar>
            <Convention>U</Convention>
```

```
            <TermConvention>U</TermConvention>
            <Rule>Forward</Rule>
          </Rules>
        </ScheduleData>
        <ApplyCoarsening>true</ApplyCoarsening>
      </Event>
      <Number>
        <Name>BarrierLevel</Name>
        <Value>0.009</Value>
      </Number>
      <Number>
        <Name>Type</Name>
        <Value>-1</Value>
      </Number>
      <Number>
        <Name>LongShort</Name>
        <Value>1</Value>
      </Number>
      <Number>
        <Name>Amount</Name>
        <Value>10000000</Value>
      </Number>
      <Currency>
        <Name>PayCcy</Name>
        <Value>USD</Value>
      </Currency>
      <Index>
        <Name>Underlying</Name>
        <Value>FX-TR20H-USD-JPY</Value>
      </Index>
    </Data>
  </ScriptedTradeData>
</Trade>
```

In the compact format the same trade looks like this:

```
<Trade id="SCRIPTED_FX_ONE-TOUCH_OPTION">
  <TradeType>ScriptedTrade</TradeType>
  <Envelope>
    <CounterParty>CPTY_A</CounterParty>
    <NettingSetId>CPTY_A</NettingSetId>
    <AdditionalFields/>
  </Envelope>
  <OneTouchOptionData>
    <Settlement type="event">2020-08-01</Settlement>
    <ObservationDates type="event">
      <ScheduleData>
        <Rules>
          <StartDate>2018-12-28</StartDate>
          <EndDate>2020-08-01</EndDate>
          <Tenor>1D</Tenor>
          <Calendar>US</Calendar>
          <Convention>U</Convention>
          <TermConvention>U</TermConvention>
          <Rule>Forward</Rule>
        </Rules>
      </ScheduleData>
```

```
      <ApplyCoarsening>true</ApplyCoarsening>
    </ObservationDates>
    <BarrierLevel type="number">0.009</BarrierLevel>
    <BarrierType type="barrierType">DownIn</BarrierType>
    <LongShort type="longShort">Long</LongShort>
    <Amount type="number">10000000</Amount>
    <PayCcy type="currency">USD</PayCcy>
    <Underlying type="index">FX-TR20H-USD-JPY</Underlying>
  </OneTouchOptionData>
</Trade>
```

The supported types that must be specified in the `type` attribute are `number`, `event`, `currency`, `dayCounter` and `index`. In addition we support some custom types that are mapped to numbers internally and allow for a more natural representation of the trade:

- `bool` with a mapping `true` $\mapsto$ 1, `false` $\mapsto$ -1
- `optionType` with a mapping `Call` $\mapsto$ 1, `Put` $\mapsto$ -1, `Cap` $\mapsto$ 1, `Floor` $\mapsto$ -1
- `longShort` with a mapping `Long` $\mapsto$ 1, `Short` $\mapsto$ -1
- `barrierType` with a mapping `DownIn` $\mapsto$ 1, `UpIn` $\mapsto$ 2, `DownOut` $\mapsto$ 3, `UpOut` $\mapsto$ 4

Arrays of events are specified as in the example above (ObservationDates), for the other types the values are listed in value tags, e.g. an array of numbers is declared as

```
  <MyNumberArray type="number">
    <Value>100.0</Value>
    <Value>200.0</Value>
    <Value>200.0</Value>
  </MyNumberArray>
```

4.4 A comment on the Payment Currency in Scripted Trades

The payoffs described in the Trade Specific Data for scripted trades usually involve a payment currency. Unless stated otherwise, this currency represents the *natural* payoff currency, i.e.

- for equity and commodity underlyings this should be the currency in which the underlying price is quoted
- for FX underlyings `FX-SOURCE-CCY1-CCY2` this should be the domestic (target, numeraire) currency `CCY2`.

If the payment currency is set to a different currency on the other hand, the resulting payoff is a true quanto payoff, i.e. the amount of the payoff is determined on the natural currency, but paid in a different currency *without* converting the amount to this latter currency using the fair FX Spot rate on the settlement date.

We are aware that a conversion from the natural payoff currency to a different settlement currency using the fair FX Spot rate is sometimes part of the terms and conditions of a trade. This conversion has no or small impact on the valuation and risk profile of a trade though and is therefore usually not part of the payoff modeling.

As an example consider a Forward Volatility Agreement on the FX pair GBP-EUR. The `PayCcy` should be set to EUR in this case, even if the forward premium is settled in GBP as it is usual market practice for this pair.

Notice that the above does not apply to fixed premiums, where the premium is given as a fixed number in the trade xml together with a premium currency in which this amount should be paid.

4.5 Payoff Scripting Language

Whitespace

Whitespace (space, tab, return, newline) is ignored during the parsing. All variable identifiers and keywords are case sensitive.

Keywords

The language uses keywords and predefined function names as listed in table 30 which may may not be used as variable identifiers.

In addition the following variable identifiers are automatically populated with special values when running the script engine on a trade:

- `TODAY`: the current evaluation date

Variables

Variables that can be used in the script are either

- externally defined variables, defined in the data node of the trade xml representation
- variables local to the script, declared within the script

Externally defined variables are protected from being modified by the script. All variables used within the script must be either externally defined or declared at the top of the script using

```
NUMBER continuationValue, exerciseValue, x[10];
```

which declares two scalars `continuationValue` and `exerciseValue` and an array `x` of size 10 (see 4.5 for more details on arrays). The only exemption to this rule is the variable declared in the `NPV` node of the script, which is definied implicitly as a scalar number.

Notice that within the script only variables of type Number can be declared.[7] All variables are initialised with 0. The scope of a variable declaration is always global to the script, multiple declarations of the same variable name are forbidden.

Variable identifiers are subject to the following restrictions

[7] this restriction allows the static analysis of the script

Keyword	Context
IF	Control Flow
THEN	
ELSE	
END	
FOR	
IN	
DO	
NUMBER	Type Identifiers
OR	Logical Operators
AND	
abs	Functions
exp	
ln	
sqrt	
normalCdf	
normalPdf	
max	
min	
pow	
black	
dcf	
days	
PAY	Model dependent functions
LOGPAY	
NPV	
NPVMEM	
HISTFIXING	
DISCOUNT	
FWDCOMP	
FWDAVG	
ABOVEPROB	
BELOWPORB	
SIZE	Other Statements
DATEINDEX	
REQUIRE	
SORT	
PERMUTE	

Table 30: Reserved keywords.

- must start with a character, then characters or numbers or underscores may follow
- no other special characters, no keywords or predefined functions allowed (see 4.5)
- e.g. `x` or `x_23`, `aValue` are valid identifiers
- identifiers starting with an underscore are technically allowed as well, but reserved for special use cases (e.g `_AMC_SimDates` and `_AMC_NPV` for AMC exposure generation)

Arrays, SIZE operator

Arrays are declared by specifying the size of the array in sqaure brackets, e.g.

```
NUMBER x[10], y[SIZE(ObservationDates)], z[5+3*v];
```

declares arrays

- `x` of size 10
- `y` with the same size as the array `ObservationDates`
- `z` with size $5 + 3v$ where v is a number variable

Once an array is delcared its size can not be changed. The ith element of an array a is accessed by `a[i]`, where i is an expression evaluating to a number. Here $i = 1, 2, 3, \ldots, n$, where n is the fixed size of the array, i.e. the subscripts start at 1 (as opposed to 0 as in some other languages).

The size of an array `a` can be evaluated by `SIZE(a)`. Only one dimensional arrays are supported. The array subscript must be *deterministic*, e.g.

```
IF Underlying > Strike THEN
  i = 1;
ELSE
  i = 2;
END;
Payoff = y[i];
```

is illegal since i in general will be path-dependent, but

```
IF Underlying > Strike THEN
  Payoff = y[1];
ELSE
  Payoff = y[2];
END;
```

is valid.[8]

[8] the background is the simplicity and performance of the engine implementation

Sorting Arrays: SORT and PERMUTE instructions

Given an array `x` of number type the statement

```
SORT (x);
```

will sort the array (pathwise) in ascending order. The statement

```
SORT (x, y);
```

will write a sorted version of `x` to `y` and leave `x` unchanged. The array `y` must be of number type and have the same size as `x`. The array `y` can also be equal to `x`, the statement `SORT(x,x);` is equivalent to `SORT(x);`. Finally the statement

```
SORT (x, y, p);
```

will write a sorted version of `x` to `y` and populate another array `p` with indices $1, \ldots,$ `SIZE`(x) such that $x[p[1]], \ldots, x[p[n]]$ is sorted. Here `p` must be an array with the same size as `x` and of number type.

A permutation `p` generated as above (or set up otherwise) can be used to sort an unrelated array `z` using

```
PERMUTE (z, p);
```

which will reorder the values of `z` as $z[1] \to z[p[1]]$, $z[2] \to z[p[2]]$ ... etc. The statement

```
PERMUTE (z, w, p);
```

will do the same, but write the result to `w` and leave `z` untouched.

Function `DATEINDEX`

Given an array `a` and a single date `d`, the expression

```
DATEINDEX(d, a, EQ)
```

returns 0 if the date d is not found in the array a and otherwise the (first) index i for which `a[i]` equals `d`. The variable d is required to be of type event. The variable a is only required to be an array, if the type of its elements are not event, the return value will always be zero indicating that d was not found in a. Similarly,

```
DATEINDEX(d, a, GEQ)
```

returns the index of the earliest date in a that is greater or equal than d, and

```
DATEINDEX(d, a, GT)
```

returns the index of the earliest date in a that is greater than d. If no such dates exists for `GEQ` or `GT`, the size of a plus 1 will be returned.

Instructions

A typical script comprises a sequence of instructions, each one terminated by `;`.

Index evaluation

Given an variable `index` of type Index its historical or projected fixing at a date d is evaluated using the expression `index(d)`. This is applicable to all index types. For example

```
Underlying(ObservationDate)
```

evaluates the index assigned to the variable Underlying at the date assigned to the variable ObservationDate. For FX, EQ, IR and COMM Spot indices this corresponds to a a fixing at the observation date in the usual sense. For COMM Future indices it is the observed future price at the observation date.

For INF indices the argument is the actual fixing date, which due to availability lags is observed at a later simulation time in models with dynamical inflation simulation. For example in the GaussianCam model, this lag is defined as the number of calendar days from the zero inflation term structure base date to its reference date (adjusted to the first date of the inflation period to be consistent with the same adjustment applied to the base date). This means that when observing an inflation index at a fixing date d, the model state at $d + \text{lag}$ is used to make this observation.

The extended syntax

```
Underlying(ObservationDate, ForwardDate)
```

evaluates the projected fixing for ForwardDate as seen from ObservationDate.

This is applicable to FX, EQ, IR, INF and COMM Spot indices, but not to COMM Future indices, since for the latter the two concepts coincide (for ForwardDate $<$ FutureExpiry). If a forward date is given for the observation of a COMM future index, no error is thrown, but it will be ignored.

For inflation indices, the ForwardDate will be the actual fixing date again and the ObservationDate will be using a lagged state as explained above.

The ForwardDate must be greater or equal than the ObservationDate. If the ForwardDate is strictly greater than the ObservationDate the ObservationDate must not be a past date (for inflation indices it must not lie before the inflation term structure's base date), since the computation of projected fixings for past dates would involve the knowledge of past curves, i.e. past market data.

Notice also the further specifics of commodity and inflation indices in 4.2.3.

Comparisons == and !=

Compares two values, e.g. `x==y` or `x!=y`. This is applicable to all types. For a number the interpretation is "numerically equal".

Comparisons <, <=, >, >=

Compares two values `x<y`, `x<=y`, `x>y`, `x>=y`. Applicable to numbers and events, but not to currencies or indices. For numbers the interpretation is "less than, but not numerically equal", "less than or numerically equal", etc.

Operations +, -, *, /

Arithmetic operations `x+y`, `x-y`, `x*y`, `x/y`, applicable to numbers only.

Assignment =

Assignment `x = y`, only allowed for numbers within the script.

Logical Operators AND, OR, NOT

Connects results of comparisons or other logical expressions:

- `x<y AND z!=0`
- `x<y OR z!=0`
- `NOT(x==y)`
- AND has higher precedence than OR, e.g.
- `x<y AND y<z OR z!=0` same as `{x<y AND y<z} OR z!=0`, but
- `x<y AND {y<z OR z!=0}` requires parenthesis
- better always use parenthesis when mixing `AND` / `OR`

Conditionals: IF ... THEN ... ELSE ...

Conditional execution can be written as

```
IF condition THEN
    ... if-body ...
ELSE
    ... else-body ...
END
```

Examples:

```
IF x == y THEN
  z = PAY(X,d,p,ccy);
  w = 1;
END;
```

```
IF x == y THEN
  z = PAY(X,d,p,ccy);
ELSE
  z = 0;
  w = 0;
END;
```

where the `ELSE` part is optional. The body can comprise one ore more instructions, each of which must be terminated by `;`.

Loops: FOR ... IN ... DO

Loops are written as

```
FOR i IN (a,b,s) DO
... body ...
END
```

where `i` is a number variable identifier, and `a`, `b`, `s` are expressions that yield a result of type Number. The variable `i` must have been declared in the script before it can be used as a loop variable. The code in the body is executed for the values $i = a, a + s, \ldots$ until $a + ks > b$ if $s > 0$ or $a + ks < b$ if $s < 0$ for some integer $k > 0$. All values a, b, s must be integers and $s \neq 0$.

Example:

```
NUMBER i,x;
FOR i IN (1,100.1) DO x = x + i; END;
```

Here `a, b` must be deterministic, `i` must not be modified in the loop body. If a or b are modified in the loop body, still the initial values read at the start of the loop are used. The loop body can comprise one or more instructions, each of which must be terminated by `;`.

Special variable: TODAY

A constant event variable, set to the current evaluation date. This can e.g. be used to restrict exercise decisions to future dates, see 4.5 for an example.

Checks: REQUIRE

If the condition `C` is not true, a runtime error is thrown. Examples:

- `REQUIRE SIZE(ExerciseDates) == SIZE(SettlementDates);`
- `REQUIRE SIZE(Underlyings) == 2;`
- `REQUIRE Strike >= 0;`

Functions `min`, `max`, `pow`

Binary functions `min(x,y)`, `max(x,y)`, `pow(x,y)`, applicable to numbers only.

Functions `-`, abs, exp, ln, sqrt

Unary functions `-x`, `abs(x)`, `exp(x)`, `ln(x)`, `sqrt(x)`, applicable to numbers only.

Functions `normalPdf`, `normalCdf`

Returns the standard normal pdf $\phi(x)$ resp. cdf $\Phi(x)$, applicable to numbers only.

Function `black`

Implements the black formula `black(omega, obs, expiry, k, f, sigma)` with

$$\begin{aligned} \text{black} &= \omega \cdot (f\Phi(\omega d_1) - k\Phi(\omega d_2)) \\ d_{1,2} &= \frac{\ln(f/k) \pm \frac{1}{2}\sigma^2 t}{\sigma\sqrt{t}} \end{aligned}$$

where t is the (model's) year fraction between obs and expiry date, i.e.:

- `omega` is 1 (call) or -1 (put)
- `obs, expiry` are the observation / expiry dates
- `k, f` are the strike and the forward
- `sigma` is the implied volatility
- notice that no discounting is applied

Function `dcf`

The expression `dcf(dc, d1, d2)` returns the day count fraction for a day count convention `dc` and a period defined by dates `d1` and `d2`.

Function `days`

The expression `days(dc, d1, d2)` returns the number of days between `d1` and `d2` for a day count convention `dc`.

Function `PAY`

The expression `PAY(X, d, p, ccy)` calculates a discounted payoff for an amount X observed at a date d, paid at a date p in currency ccy, i.e.

$$\frac{XP_{ccy}(d,p)FX_{\text{ccy,base}}(d)}{N(d)} \tag{1}$$

where

- here P_{ccy} is the discount factor in currency ccy, FX is the FX spot from ccy to base and N is the model numeraire
- $d \leq p$ must hold
- if p lies on or before the evaluation date, the result is zero; X is not evaluated in this case. Note that X is evaluated in the LOGPAY function if past cashflows are included, see 4.5.
- avoids reading non-relevant past fixings from the index history
- if d lies before (but p after) the evaluation date, it is set to the evaluation date, i.e. the result is computed as of the evaluation date

Function `LOGPAY`

The expression `LOGPAY(X, d, p, ccy)` has the same meaning as `PAY(X, d, p, ccy)` (see 4.5) but as a side effect populates an internal cashflow log that is used to generate expected flows. The generated flow is

$$\frac{N(0)E\left(\frac{XP_{ccy}(d,p)FX_{\text{ccy,base}}(d)}{N(d)}\right)}{FX_{\text{ccy,base}}(0)P_{ccy}(0,p)} \tag{2}$$

which ensures that the flows discounted on T0 curves and converted with T0 FX Spots reproduce the NPV generated from `LOGPAY` expressions.

There is a second form `LOGPAY(X, d, p, ccy, legNo, type)` taking in addition

- a leg number `legNo`, which must evaluate to a determinisitc number
- a cashflow type `type`, which is an arbitrary string meeting the conventions for variable names

This additional information is used to populate the ORE cashflow report. If not given, `legNo` is set to 0 and type is set to `Unspecified`. Notice that cashflows will equal pay dates, pay currencies, leg numbers and types are aggregated to one number in the cashflow report.

A third form `LOGPAY(X, d, p, ccy, legNo, type, slot)` takes an additional parameter `slot` which must evaluate to a whole positive and deterministic number $1, 2, 3, \ldots$. If several cashflows are logged into the same slot, previous results are overwritten. This is useful for scripts where tentative cashflows are generated that are later on superseded by other cashflows (e.g. for an American option).

Examples for the three forms are given below:

```
Payoff1 = LOGPAY( Notional * fixedRate, PayDate, PayDate, PayCcy);
Payoff2 = LOGPAY( Notional * fixedRate, PayDate, PayDate, PayCcy, 2, Interest);
Payoff3 = LOGPAY( Notional * fixedRate, PayDate, PayDate, PayCcy, 2, Interest, 3);
```

Here, Payoff1 will appear under leg number 0 and flow type "Unspecified" in the cashflow report. Payoff2 will appear under leg number 2 and flow Type "Interest". The

same holds for Payoff3, but if any amounts were booked using the slot parameter 3 previously they will be overwritten with the current amount.

Note: If IncludePastCashflows in the pricing engine config is set to true then even if p lies on or before the evaluation date, a cashflow entry will be generated.

Function `NPV, NPVMEM`

The expression `NPV(X, d, [C], [R1], [R2])` calculates a conditional NPV of an amount X conditional on a date d, i.e.

$$E(X \mid \mathcal{F}_d \cap \mathcal{F}_C) \tag{3}$$

where $\mathcal{F}_d$ is the sigma algebra representing the information generated by the model up to d and $\mathcal{F}_C$ represents the additional condition C (if given). In an MC model 3 is computed using a regression against the model state at d. C can be used to filter the training paths, e.g. on ITM paths only. d must not lie before the evaluation date, but for convenience the scipt engines will treat d as if it were equal to the evaluation date in this case for the purpose of the NPV function evaluation.

The regressor can be enriched by (at most 2) additional variables R_i. A typical usage is the accumulated coupon in a target redemption feature which heavily influences the future conditional NPV but is not captured in the model state.

A typical usage of the NPV function is to decide on early exercises in the Longstaff-Schwartz algorithm:

```
NUMBER Payoff, d;
FOR d IN (SIZE(ExerciseDates), 1) DO
    IF ExerciseDates[d] > TODAY THEN
      Payoff = PAY( PutCall * (Underlying(ExerciseDates[d]) - Strike),
                    ExerciseDates[d], ExerciseDates[d], PayCcy);
      IF Payoff > 0 AND Payoff > NPV( Option, ExerciseDates[d], Payoff > 0) THEN
        Option = Payoff;
      END;
    END;
END;
Option = LongShort * Quantity * Option;
```

Here `TODAY` represents the evaluation date to ensure that only future exercise dates are evaluated, see 4.5.

Note: It is the users responsibility to use NPV() correctly to a certain extend: An example would be that X is composed from both past and future fixings w.r.t. the observation time t. In that case only the future fixings should be included in the argument of NPV(), whereas the past fixings are known and should just be added to the result of NPV().

The variant `NPVMEM(X, d, s, [C], [R1], [R2])` works exactly like `NPV(X, d, [C], [R1], [R2])` except that it takes an additional parameter `s` that must be an integer. If `NPVMEM()` is called more than once for the same parameter `s` a regression model representing the conditional npv will only be trained once and after that the trained

model will be reused. The usual use case is for scripts used in combination with the AMC module where a regression model will be trained on a relative large number of paths (specified in the pricing engine configuration) and then reused in the global exposure simulation on a relatively small number of paths (specified in the xva simulation setup).

Function `HISTFIXING`

The expression `HISTFIXING(Underlying, d)` returns 1 if d lies on or before the reference date *and* the underlying has a historical fixing as of the date d and 0 otherwise.

Function `DISCOUNT`

The expression `DISCOUNT(d, p, ccy)` calculates a discount factor $P_{ccy}(d, p)$ as of d for p in currency *ccy*. Here d must not be a past date and $d \leq p$ must hold.

Functions `FWDCOMP` and `FWDAVG`

The `FWDCOMP()` and `FWDAVG()` functions are used to calculate a daily compounded or averaged rate over a certain period based on an overnight index such as USD-SOFR, GBP-SONIA, EUR-ESTER etc..

The rate is estimated as seen from an observation date looking *forward* from that date, even if fixings relevant for the rate lie in the past w.r.t. the observation date. In the latter case, an approximation to the true rate which is then dependent on the path leading from TODAY to the current model state at the observation date is calculated. This approximation is model-dependent. The only exception to this mechanics are historical fixings that are *known* as of TODAY. Such fixings are always taken into consideration with their true value.

More specifically, the `FWDCOMP()` and `FWDAVG()` functions take the following parameters. The parameters must be given in that order, and all parameters must be given in sequence up to the parameter "end" (last mandatory parameter) or the end of an optional parameter group (i.e. an optional parameter group must be given as a whole). Furthermore, all parameters must be deterministic.

- index [mandatory]: an overnight index `index`, e.g. EUR-EONIA, USD-SOFR, ...
- obs [mandatory]: an observation date obs $\leq$ start; if obs is $<$ TODAY it is set to TODAY, i.e. the result is as of TODAY in this case
- start [mandatory]: the value start date, this might be modified by a non-zero lookback
- end [mandatory]: the value end date, this might be modified by a non-zero lookback
- spread [optional group 1]: a spread, defaults to 0 if not given
- gearing [optional group 1]: a gearing, defaults to 1 if not given

- lookback [optional group 2]: a lookback period given as number of days, defaults to 0 if not given. This argument must be given as either a constant number or a plain variable, i.e. not as a more complex expression than either of these.
- rateCutoff [optional group 2]: a rate cutoff given as number of days, defaults to 0 if not given
- fixingDays [optional group 2]: the fixing lag given as number of days, defaults to 0 if not given. This argument must be given as either a constant number or a plain variable, i.e. not as a more complex expression than either of these.
- includeSpread [optional group 2]: a flag indicating whether to include the spread in the compounding, a value equal to 1 indicates 'true', -1 false, defaults to 'false' if not given
- cap [optional group 3]: a cap value, defaults to 999999 (no cap) if not given
- floor [optional group 3]: a floor value, defaults to -999999 (no floor) if not given
- nakedOption [optional group 3]: a flag indicating whether the embedded cap / floor should be estimated, a value equal to -1 indicates 'false' (capped / floored coupon rate is estimated), 1 'true' (embedded cap / floor rate is estimated), defaults to 'false' if not given
- localCapFloor [optional group3]: a flag indicating whether the cap / floor is local, a value equal to -1 indicates 'false', 1 'true', defaults to 'false' if not given.

Based on these parameters a rate corresponding to that computed for a vanilla floating leg is estimated, see the description in section "Floating Leg Data, Spreads, Gearings, Caps and Floors" for more details on this.

Functions `ABOVEPROB, BELOWPROB`

These functions are only available in Monte Carlo engines. The expression

```
ABOVEPROB(underlying, d1, d2, U)
```

returns the pathwise probability that the value of an index `underlying` lies at or above a number U for at least one time t between dates $d1$ and $d2$ conditional on the underlying taking the simulated path values at $d1$ and $d2$. The probability is by definition computed assuming a continuous monitoring. Similarly,

```
BELOWPROB(underlying, d1, d2, D)
```

returns the probability that the value of the underlying lies at or below D. Notice that $d1$ and $d2$ should be adjacent simulation dates to ensure that the results computed in the script are meaningful. This means the script should not evaluate the underlying at a date d with $d1 < d < d2$.

We note that U and D are not required to be deterministic quantities, although the common use case will probably be to have path-independent inputs.

Finally, if $d1 > d2$ both functions return 0.

5 Pricing Methodology

The following table provides an overview over the pricing approaches taken per product type. The subsequent sections elaborate on these approaches for groups of products by asset class.

Asset Class	Product Type	Pricing Model	Calibration
IR	Interest Rate Swap	Discounted Cashflows	
IR	Forward Rate Agreement	Discounted Cashflows	
IR/FX	Cross Currency Swap	Discounted Cashflows	
IR	OIS Swap	Discounted Cashflows	
IR	Zero Coupon Swap	Discounted Cashflows	
IR	BMA Swap	Discounted Cashflows	
IR	CMS Swap, Cap/Floor	Linear Terminal Swap Rate Model (LTSR)	
IR	Cap, Floor Collar on IBOR	Black Model for log-normal and shifted log-normal volatilities; Bachelier Model for normal volatilities	Smile used
IR	European Vanilla Swaption	Black Model for log-normal and shifted log-normal volatilities; Bachelier Model for normal volatilities	Smile used
IR	Bermudan Vanilla Swaption	Linear Gauss Markov (LGM)	Smile used
IR	Cap, Floor Collar on CMS	Linear Terminal Swap Rate Model (LTSR)	Smile used
IR	European CMS Spread Option	Log-Normal Swap Rate Model	Smile used
IR	Flexi Swap	Jamshidian Replication, Linear Gauss Markov	Smile used
IR	Balance Guaranteed Swap	as Flexi Swap	Smile used
IR	Bond Option	Black Model	Smile used
IR	Swap with Formula-based Coupon	Log-Normal Swap and Ibor Rate Model	Smile used
FX	FX Forward	Discounted Cashflows	
FX	FX Swap	Discounted Cashflows	
FX	European FX Option	Garman-Kohlhagen Model	Smile used
FX	American FX Option	Finite Difference Methods with the Black-Scholes framework	Smile used
FX	FX Barrier Option	Garman-Kohlhagen Model with analytic formulas	Smile used
FX	Digital FX Option	Garman-Kohlhagen Model with analytic formulas	Smile used
FX	Digital FX Barrier Option	Finite Difference Methods with the Black-Scholes framework	Smile used
FX	FX Touch Option	Garman-Kohlhagen Model with analytic American engine	Smile used
FX	FX Double Touch Option	Garman-Kohlhagen Model with analytic formulas	Smile used
FX	FX Double Barrier Option	Garman-Kohlhagen Model with analytic formulas	Smile used
FX	FX Variance and Volatility Swap	Replicating portfolio method	Smile used
INF	CPI Swap	Discounted Cashflows	
INF	YoY Inflation Swap	Discounted Cashflows	
EQ	Equity Forward	Discounted Cashflows	

EQ	Equity Barrier Option	Black Model	Smile used
EQ	Equity Double Barrier Option	Black Model	Smile used
EQ	Equity Swap	Discounted Cashflows	
EQ	European Equity Option	Black Model	Smile used
EQ	Quanto European Equity Option	Black Model	Smile used
EQ	Equity Variance Swap	Replicating portfolio method	Smile used
FX/EQ	Double Digital Option	Black Model	ATM
FX/EQ	Autocallable Type 01	Black Model	ATM
FX/EQ	Performance Option Type 01	Black Model	ATM
FX/EQ	Window Barrier Option	Black Model	ATM
FX/EQ	Target Redemption Forward (TaRF)	Black Model	ATM
CR	Credit Default Swap	Default Curves, Discounted Cashflows	
CR	CDS Option	Black Model	
CR	Synthetic CDO	Gauss Copula and Hull-White Bucketing	
CR	Risk Participation Agreement	European Swaption Representation or LGM Grid	
CR	Credit Linked Swap	Discounted Cashflows	
COM	Commodity Forward	Discounted Cashflows	
COM	European Commodity Option	Black Model	Smile used
COM	Commodity Swap	Discounted Cashflows	
COM	Commodity Basis Swap	Discounted Cashflows	
COM	Commodity Swaption	Black Model	
COM	Commodity Average Price Option	Black Model	
HYB	Composite Trade	Deferred to component trades	

5.1 Interest Rate Derivatives

Vanilla IR instruments are priced using a multi-currency framework of discount curves by cash flow currency and forwarding curves per index (see Section 7 for details of curve building).

If the floating leg of an interest rate swap contains capped/floored/collared payments, the pricing approach described in the relevant cap/floor instrument methodology is applied to the cap/floor components.

5.1.1 Interest Rate Swap

The present value of a Vanilla Interest rate Swap receiving fixed and paying variable interest is

$$NPV = NPV_{fix} - NPV_{flt}$$

with fixed leg NPV

$$NPV_{fix} = \sum_{i=1}^{n^{fix}} N_i^{fix}\, c_i\, \delta_i^{fix}\, P(t_i^{fix})$$

and floating leg NPV

$$NPV_{flt} = \sum_{i=1}^{n^{flt}} N_i^{flt} \left(f_i + s_i\right) \delta_i^{flt} \, P(t_i^{flt})$$

and

- $n^{fix/flt}$: the number of coupons for the fixed resp. floating leg - note that period length/tenor generally differ between legs
- $N_i^{fix/flt}$: fixed resp. floating leg notional for period i, possibly constant
- c_i: the fixed rate for period i, possibly constant
- f_i: the forward projected IBOR rate for period from t_{i-1}^{flt} to t_i^{flt} - note that for the current period, this will be a fixing instead of a forward rate (assuming fixing in advance)
- s_i: spread for floating leg period i
- δ_i^{fix}: the daycount fraction in years for the fixed leg period i from t_{i-1}^{fix} to t_i^{fix}
- δ_i^{flt}: the daycount fraction in years for the floating leg period i from t_{i-1}^{flt} to t_i^{flt}
- $P(t)$: the discount factor for time t

Note that the forward rate $f(t_{i-1}, t_i)$ above is derived from a forward curve $P^f(t)$ via

$$f(t_{i-1}, t_i) \, \delta(t_{i-1}, t_i) = \frac{P^f(t_{i-1})}{P^f(t_i)} - 1$$

where the forward curve $P^f(t)$ is generally different from the discount curve $P(t)$, see section 7.

5.1.2 Forward Rate Agreement

The NPV of a Forward Rate Agreement, from the point of view of the party receiving the fixed predetermined rate, assuming payments are made at time t_2:

$$NPV = N \cdot (c - f(t_1, t_2)) \cdot \delta(t_1, t_2) \cdot P(t_2)$$

where:

- c: the predetermined fixed forward interest rate, set in the terms of the FRA (at t_0) to be received at time t_2
- $f(t_1, t_2)$: the forward projected IBOR rate for the period from t_1 to t_2, to be paid at t_2
- $\delta(t_1, t_2)$: the daycount fraction in years for the period from t_1 to t_2
- $P(t_2)$: the discount factor for time t_2

Note that market practice dictates the payments are made in t_1 by computing the cash flow in t_2 and discounting it to t_1. For pricing purposes, this has practically no impact - see (Lichters, Stamm, & Gallagher, 2015) Ch. 11.1.

5.1.3 Single Currency Basis Swap

The Single Currency Basis Swap exchanges legs (A and B) of variable payments with generally different index tenors and payment frequencies, so that we can write the present value as

$$NPV = NPV_A - NPV_B$$

$$NPV_A = \sum_{i=1}^{n^A} N_i^A \left(f_i^A + s_i^A\right) \delta_i^A \, P(t_i^A)$$

$$NPV_B = \sum_{i=1}^{n^B} N_i^B \left(f_i^B + s_i^B\right) \delta_i^B \, P(t_i^B)$$

where

- f_i^A: the forward projected rate for period i from t_{i-1}^A to t_i^A for IBOR index A and tenor A - this will be a fixing instead of a forward rate for the current period
- f_i^B: the forward projected rate for period i from t_{i-1}^B to t_i^B for IBOR index B and tenor B - this will be a fixing instead of a forward rate for the current period
- $s_i^{A,B}$: spread for floating leg A's resp. B's period i
- $\delta_i^{A,B}$: the daycount fraction in years for leg A's resp. B's period i from $t_{i-1}^{A,B}$ to $t_i^{A,B}$

5.1.4 Cross Currency Swap

Cross Currency (XCCY) Swaps are priced using a multi-currency framework of discount curves by cash flow currency and forwarding curves per index, and generally have notional exchanges at the beginning and the end of the swap set to reflect the prevailing FX spot rate on trade date. However, XCCY Swaps may also be rebalancing, having "FX resets" (i.e. notional adjustments at each fixing date to compensate the FX rate move since last fixing). This feature has an impact on discount curve building (Lichters, Stamm, & Gallagher, 2015) Ch 4.3 (see Section 7.5 for more details on curve building).

If a Cross Currency Swap contains capped/floored/collared payments, the pricing approach described in the relevant cap/floor instrument methodology is applied to the cap/floor components.

The present value of a non-rebalancing XCCY Swap (both legs floating, constant notional) receiving currency A and paying currency B can be expressed as follows:

Base currency NPV

$$NPV = X_A \cdot NPV_A - X_B \cdot NPV_B$$

with leg A NPV in currency A

$$NPV_A = -N^A \, P^A(t_0) + N^A \sum_{i=1}^{n_A} (f_i^A + s_i^A) \, \delta_i^A \, P^A(t_i^A) + N^A \, P^A(t_n^A)$$

and leg B NPV in currency B

$$NPV_B = -N^B\, P^B(t_0) + N^B \sum_{i=1}^{n_B} (f_i^B + s_i^B)\, \delta_i^B\, P^B(t_i^B) + N^B\, P^B(t_n^B)$$

where:

- X_A: the FX spot rate between currency A and the base currency at valuation time
- X_B: the FX spot rate between currency B and the base currency at valuation time
- N^A: the notional of the currency A leg expressed in currency A
- N^B: the notional of the currency B leg expressed in currency B
- $P^A(t_i^A)$: the discount factor for time i for currency A
- $P^B(t_i^B)$: the discount factor for time i for currency B
- $s_i^{A,B}$: the contractual XCCY basis swap spread on leg A resp. B for period i
- f_i^A: the forward projected IBOR rate for period i for the currency A leg
- f_i^B: the forward projected IBOR rate for period i for the currency B leg

We are assuming here that the date of the initial exchange of notionals has not passed yet. Once passed, these cash flows - like any past flows - are excluded from valuation. And moreover, the forward projected rates for both legs will be fixings for the current period, assuming fixings in advance.

The Swap shown above, exchanging floating for floating interest, is a Cross Currency Basis Swap (CCBS). A XCCY Swap may as well exchange fixed for floating or fixed for fixed payments.

Rebalancing Cross Currency Swap

The PV for each leg of a rebalancing XCCY Basis Swap between currency A and currency B, where the rebalancing happens on leg B, can be expressed as follows:

$$NPV = X_A \cdot NPV_A - X_B \cdot NPV_B$$

$$NPV_A = -N^A\, P^A(t_0) + N^A \cdot \sum_{i=1}^{n_A} (f_i^A + s_i^A)\, \delta_i^A\, P^A(t_i^A) + N^A\, P^A(t_n^A)$$

$$NPV_B = -N_0^B\, P^B(t_0) + \sum_{i=1}^{n_B} \left(N_{i-1}^B (f_i^B + s_i^B)\, \delta_i^B + (N_{i-1}^B - N_i^B)\right)\, P^B(t_i^B) + N_{n_B}^B\, P^B(t_{n_B}^B)$$

The rebalancing feature causes intermediate notional cash flows $(N_{i-1}^B - N_i^B)$ on leg B. The future leg B notional amounts are computed using FX Forward rates

$$N_i^B = N^A \cdot \frac{X_A(t_i)}{X_B(t_i)}$$

such that the future leg B expected notional converted into leg A currency will correspond to the constant leg A notional. This approach assumes zero correlation

between the relevant interest and FX rates. Non-zero correlation would lead to a convexity adjustment, see e.g. [46] section 12.5.

5.1.5 Overnight Index Swap

An OIS Swap has a floating leg that compounds on a daily basis using an overnight index. The present value of a floating OIS leg can be calculated by summing the NPVs for each coupon payment including spread (if any). Flat compounding is implemented for OIS Swaps (i.e. the spread does not compound). The OIS leg's present value can be written

$$NPV_{OIS} = N \sum_{i=1}^{n} (f_i + s_i)\, \delta_i\, P_{OIS}(t_i^p)$$

where

- N: notional, assumed constant
- f_i: forward rate for future starting period i computed as

$$f_i = \frac{1}{\delta_i} \left(\frac{P_{OIS}(t_i^s)}{P_{OIS}(t_i^e)} - 1 \right)$$

- $t_i^{s,e,p}$: the value dates associated to the first and last fixing dates in period i, and the payment date of period i
- δ_i: year fraction of period i
- $P_{OIS}(t)$: the discount factor for time t bootstrapped from the OIS curve
- s_i: the spread applicable to coupon i

When the valuation date t falls into the accrual period then we replace the forward f_i as follows, to take the compounded accrued interest from period start to today into account:

$$f_i = \frac{1}{\delta_i} \left[\prod_{j=0}^{j=m-1} \left(1 + ON(t_j^f)\, \delta(t_j^f, t_{j+1}^f) \right) \frac{P_{OIS}(t_m^v)}{P_{OIS}(t_i^e)} - 1 \right]$$

where $ON(t_j^f)$ is the overnight fixing for fixing date t_j^f with t_0^f, ..., t_m^f denoting the fixing dates in period i with associated value dates t_j^v. Here t_m^f is either equal to today if today's fixing is known or to the last fixing date before today if today's fixing is not known.

The payment frequency on OIS legs with maturities up to a year is typically once at maturity, for maturities beyond a year it is annually.

In an Overnight Index Swap one typically swaps an OIS leg for fixed leg.

US Overnight Index Swap

In the US OIS market another flavour of Overnight Index Swaps is more common, where an OIS leg is exchanged for a floating, USD-LIBOR-3M linked, leg. Moreover, the OIS leg is not a compounded OIS leg as discussed above, but the OIS leg pays a weighted arithmetic average of overnight (FedFunds) fixings quarterly, see e.g. [51].

Apart from conventions, this changes the accrued interest calculation for the case when the valuation date t falls into coupon period i to

$$f_i = \frac{1}{\delta_i}\left[\sum_{j=0}^{j=m-1} ON(t_j^f)\,\delta(t_j^f, t_{j+1}^f) + \ln\left(\frac{P_{OIS}(t_m^v)}{P_{OIS}(t_i^e)}\right)\right]$$

using Takada's approximation for efficient forecasting from today to the coupon period end, see [51], formula 3. As in 5.1.5 t_m^f is either equal to today or the last fixing date before today depending on whether today's fixing is known.

5.1.6 Zero Coupon Swap

On compounding swaps, the floating rate is applied to the notional amount plus accrued interest caused by the previous floating rate(s). On payment date, the accrued amounts for multiple accrual periods are summed. Both straight[9] and flat compounding is supported for zero-coupon swaps.

A zero-coupon swap compounds at the tenor of the swap and pays only at maturity. There are zero-coupon fixed and zero-coupon floating legs:

$$NPV = P(T)\sum_{i=1}^{n} A_i$$

$$A_i = \begin{cases} N\,c\,\delta_i + \left(\sum_{j=1}^{i-1} A_j\right) c\,\delta_i & ; \text{ for a fixed ZC leg} \\ N\,(f_i + s_i)\,\delta_i + \left(\sum_{j=1}^{i-1} A_j\right)(f_i + s_i)\,\delta_i & ; \text{ for a floating ZC leg} \end{cases}$$

where:

- N: notional
- c: the fixed rate
- f_i: the forward projected IBOR rate for period i from t_{i-1} to t_i
- s_i: floating spread for period i
- δ_i: the daycount fraction in years for the period from t_{i-1} to t_i
- $P(T)$: the discount factor for maturity T
- A_i: the accrued interest up to period u

5.1.7 BMA Swap

A BMA Swap leg pays the un-compounded arithmetic average of weekly BMA Index fixings monthly. The present value of a BMA Swap leg can be expressed as follows:

$$NPV = N\sum_{i=1}^{n}(f_i + s_i)\,\delta_i\,P(t_i)$$

Where:

[9] The spread compounds (i.e. is applied to the notional amount plus accrued interest).

- N: notional amount, typically constant, hence we have taken the notional outside the sum above
- f_i: the forward projected BMA rate for period i from t_{i-1} to t_i, composed of the arithmetic average of projected weekly BMA fixings (where the current week is already fixed for the current period)
- s_i: spread for period i
- δ_i: the daycount fraction in years for the period from t_{i-1} to t_i
- $P(t_i)$: the discount factor for maturity t_i

5.1.8 CMS Swap, CMS Cap/Floor

CMS-linked cash flows (in CMS Swaps and CMS Cap/Floors) are priced using replication approaches following Andersen & Piterbarg's Linear Terminal Swap Rate model (LTSR), see section 6.3. The LTSR model supports normal, log-normal, and shifted log-normal volatilities. LTSR model parameters are Mean Reversion, and upper and lower rate bounds.

5.1.9 CMB Swap

Constant Maturity Bond (CMB)-linked coupons (linked to an index of form CMB-FAMILY-TENOR) are priced using simple projection of a bond yields, i.e. computing the yield of a forward starting bond with start date on the coupon's fixing date and same tenor as specified in the index' name. The forward yield calculation is determined by the reference data maintained for the Bond family, its associated yield curve (stripped from benchmark bonds prices or yields of that family) and the conventions for converting bond prices into yields.

Note that in contrast CMS index projections, we currently do not apply convexity adjustments to projected CMB indices.

5.1.10 Digital CMS Option

Similar to CMS swaps, Digital CMS Options are priced using the Linear Terminal Swap Rate model (LTSR) (see section 6.3), while the digital coupon payoffs in the underlying CMS swap are replicated using call/put option spreads.

5.1.11 Cap/Floor

Interest rate caps, floors, and collars are supported for underlyings of type

- Ibor (e.g. USD-LIBOR-3M)
- USD-SIFMA
- forward looking RFR term rate, e.g. USD-SOFR-3M
- backward looking RFR rates, e.g. USD-SOFR compounded over 3M periods

We use the Black model for log-normal and shifted log-normal volatilities, and the Bachelier model for normal volatilities. To price a cap which is a series of caplets we price the individual caplets and add up the NPVs of the caplets[10].

5.1.12 Pricing of an Ibor or forward looking RFR term rate caplet

A caplet on an Ibor or forward looking RFR term rate $F(t, t_s, t_e)$ as seen from t with fixing time $t_f > t \geq 0$, index rate projection period $[t_s, t_e]$ and pay time t_p has the price ν_F

$$\nu_F = N \cdot P(0, t_p) \cdot \text{Black/Bachelier}(K, F(0, t_s, t_e), \sigma(t_f)\sqrt{t_f}, \omega) \tag{4}$$

where "Black/Bachelier" refers to the well-known pricing formulas 6.2 resp. 6.1 and

- N is the notional of the caplet
- $P(0, t_p)$ is the applicable discount factor, usually the discount factor on an overnight curve corresponding to the collateralization of the cap
- K is the strike of the caplet
- $F0, t_s, t_e)$ is the projected forward rate as seen from the valuation time $t = 0$
- $\sigma = \sigma(t_f)$ is the volatility of the caplet. The volatility is read from the interpolated caplet volatility surface for the underlying Ibor / term rate index at time t_f and strike K. The caplet volatility surface is either
  - bootstrapped from quoted market volatilities as described in 5.1.15 or
  - proxied by a another caplet volatility surface as described in 5.1.16. This is useful if no direct market quotes are available for the underlying index. For example, we might use liquid market quotes to boostrap a volatility surface for backward looking USD-SOFR caplets and use the resuling caplet surface to proxy caplet volatilities for forward looking USD-SOFR-3M term rates.
- $\sigma^2 t_f$ is the accrued variance from $t = 0$ to the fixing time t_f of the caplet. The square root of the accrued variance is the input to the Black / Bachelier model (6.2, 6.1) and called "terminal volatility" in the context there
- $\omega \in \{-1, +1\}$ is 1 for a caplet and -1 for a floorlet

We mention that if the caplet uses a non-standard timing, i.e. a fixing time t_f such that $t_e \neq t_p$, the forward rate $F(0, t_s, t_e)$ entering the Black / Bachelier formula 4 is replaced by a timing-adjusted forward rate. We do not go into further details on this point here.

5.1.13 Pricing of an backward looking RFR caplet

The general framework to price backward looking RFR caplet is outlined in [65]. In the Forward Market Model (FMM) we consider the term rate R spanning the accrual period $[S, T]$ with length τ

[10] Here and in what follows we often refer to "cap" and "caplet" but mean both caps / floors resp. caplets / floorlets

$$R(S,T) := \frac{1}{\tau}\left(e^{\int_S^T r(u)du} - 1\right) \tag{5}$$

$R(S,T)$ is the continuous version of the daily compounded rate. Denote $R(T_{j-1},T_j)$ as seen from t as $R_j(t)$ on a forward rate grid $T_0,\ldots,T_n$, then $R_j(t)$ is a martingale in the T-forward measure and gives rise to the (normal) FMM dynamics

$$dR_j(t) = \sigma_j(t)g_j(t)dW_j(t) \tag{6}$$

where

- $\sigma_j(\cdot)$ is an adapted process
- g is a scaling function that accounts for the decreasing volatility of R_j in $[T_{j-1},T_j]$:
  - g is piecewise differentiable (technical requirement)
  - $g(t) = 1$ for $t < T_{j-1}$, i.e. no scaling before the start of the accrual period of $R(t,T_j)$
  - $g(t)$ is monotonically decreasing in $[T_{j-1},T_j]$
  - $g(t) = 0$ for $t \geq T_j$
  - In the following we use a linear decay: $g_j(t) = \min\left(\frac{(T_j-t)^+}{T_j-T_{j-1}},1\right)$

We note that It is straightforward to modify 6 to use lognormal or shifted lognormal dynamics. We do not go into further details here.

In the FMM framework, a (backward looking) cap with strike K has payoff

$$\max(\omega(R_j(T_j)-K),0) \tag{7}$$

Assuming $\sigma_j(\cdot)$ to be deterministic we get a Bachelier formula for the caplet price ν_B

$$\nu_B = N\cdot P(0,T_j)\cdot \text{Black/Bachelier}(K,R_j(0),v,\omega) \tag{8}$$

where the variance v^2 is given by

$$v^2 = \int_0^{T_j} \sigma_j(s)^2 g(s)^2 ds \tag{9}$$

We use the explicit formula for v^2 from [65] in section 6.3

$$v^2 = \sigma_j^2\left[T_{j-1}^+ + \frac{1}{3}\frac{(T_j - T_{j-1}^+)^3}{(T_j-T_{j-1})^2}\right] \tag{10}$$

Summarizing, this leads to the following price ν_B for a backward looking RFR caplet:

$$\nu_B = N \cdot P(0, t_p) \cdot \text{Black/Bachelier}(K, F(0, t_s, t_e), v, \omega) \tag{11}$$

where

- N is the notional of the caplet
- $P(0, t_p)$ is the applicable discount factor for the payment time t_p
- K is the strike of the caplet
- $F(0, t_s, t_e)$ is the projected forward rate as seen from $t = 0$ for the compounding period $[t_s, t_e]$.
- v^2 is the accrued variance from $t = 0$. As above, the square root of the accrued variance, i.e. v is the input to the Black / Bachelier model (6.2, 6.1) and called "terminal volatility" there. Following the derivation above, we have

$$v^2 = \sigma(t_0)^2 \left[t_0^+ + \frac{1}{3} \frac{(t_1 - t_0^+)^3}{(t_1 - t_0)^2} \right] \tag{12}$$

  with

  - t_0, t_1 denoting the first and last relevant fixing times of the overnight fixings underlying the compounded RFR rate. Notice that t_0 might be negative, i.e. part of the underlying overnight fixings are already known.
  - $\sigma(t_0)$ is the volatility of the RFR caplet. This is read from the interpolated caplet volatility surface for the underlying Ibor / term rate index at time t_0 and strike K (we set $\sigma(t) = \sigma(0)$ for $t < 0$).

- $\omega \in \{-1, +1\}$ is 1 for a caplet and -1 for a floorlet

Notice the important difference between $\sigma(t_0)$ and the effective Black / Bachelier caplet volatility $\sigma_{\text{eff}}(t_1)$ which can be defined by

$$\sigma_{\text{eff}}(t_1)^2 t_1 = v^2 \tag{13}$$

and which allows to price an RFR caplet as if it was a standard Ibor caplet with caplet volatility $\sigma_{\text{eff}}(t_1)$ and fixing time t_1. We do not use σ_{eff} to represent caplet volatilities in bootstrapped or proxied caplet volatility surfaces for RFR underlyings. We also do not report σ_{eff} as the pricing vol in the cashflow report. Instead we use $\sigma(t_0)$ in all cases (where we set $\sigma(t_0) := \sigma(0)$ for $t_0 < 0$).

5.1.14 Pricing of a SIFMA caplet

The methodology to price a SIFMA caplet is similar to a backward looking arithmetic average overnight caplet.

5.1.15 Bootstrap of caplet volatilities

Quoted par market volatilities for caps are not directly used for pricing. Instead we build an interpolated caplet surface with nodes

$$\sigma(t_i, K_j) \tag{14}$$

for caplet expiries t_i, $i = 1, \ldots, n$ and strikes K_j, $j = 1, \ldots, m$. The caplet volatilities $\sigma(t_i, K_j)$ are determined in a bootstrap procedure such that the premiums of the quoted market caps are reproduced. The pricing of the caplets both on the interpolated caplet surface and to generate the benchmark market premiums follow the methods outlined in 4 and 11 for Ibor / forward looking term rates and backward looking RFR rates. We further note:

- for Ibor / forward looking term rates the first caplet of a cap is excluded for the purpose of the bootstrap
- for backward looking RFR rates the first caplet of a cap is included for the purposes of the bootstrap
- the bootstrapped volatility for an Ibor / forward looking term rate caplet is stored at node (t_f, K) where t_f is the fixing time of the caplet and K is the strike.
- the bootstrapped volatility for a backward looking RFR caplet is stored at node (t_0, K) where t_0 is the fixing time of the first relevant overnight fixing entering the compounded RFR rate and K is the strike. The stored volatility is the value $\sigma(t_0)$ in the sense of the RFR caplet pricing model description above (i.e. it is *not* the effective volatility σ_{eff} defined in 13).

5.1.16 Volatility Proxies for caps

For underlyings for which no direct market volatility quotes exist, a proxy caplet volatility surface can be defined. The proxy surface is based on a bootstrapped caplet volatility surface for another underlying. For example,

- a USD-SOFR-3M term rate caplet volatility surface can be proxied by a USD-SOFR backward looking RFR caplet surface with 3M compounding periods, where the latter is bootstrapped from market quotes
- a USD-SOFR backward looking RFR caplet surface with 3M compounding periods can be proxied by a USD-LIBOR-3M Ibor caplet volatility surface
- etc. ... all combinations of Ibor / forward looking and RFR backward looking surface types are allowed to derive a proxy caplet surface from a bootstrapped surface

When deriving a proxy surface, the base surface is bootstrapped first and the proxy surface is based on the resulting caplet volatilities of the base surface. The volatility $\sigma_P(t, K)$ for a time t and strike K on the proxy surface is then given as

$$\sigma_P(t, K) = \sigma_B(t, K + \Delta K)$$

where $\sigma_B(\cdot)$ is the caplet volatility function on the base surface and ΔK is a strike adjustment. The strike adjustment is computed as

$$F_B(t) - F_P(t) \tag{15}$$

where F_B and F_P are the fair forward rates for the base resp. proxy underlying at the "anchor time" t, which is given as

- the fixing time of the unique fixing date for an Ibor / forward looking RFR term rate
- the time of the first relevant overnight fixing for an backward looking RFR rate

For example, if the base surface is a USD-SOFR backward looking RFR caplet volatility surface (with 3M compounding periods) and the proxy surface is a USD-SOFR-3M term rate surface, the volatility σ_T for a USD-SOFR-3M term rate caplet for strike K with and fixing time t_f and fair rate F_T will be given as

$$\sigma_R(t_f, K + (F_R - F_T)) \tag{16}$$

where F_R is the fair backward looking USD-SOFR rate for the period $[t_f, t_f + 3M]$ and $\sigma_R(\cdot)$ is the volatility on the backward looking RFR caplet surface.

5.1.17 European Swaption

Standard European Swaptions are priced using the Black model for log-normal and shifted log-normal volatilities, and the Bachelier model for normal volatilities. The full volatility smile (cube) is taken into account where available.

Trades are considered a "standard European swaption" if the following criteria are met:

- the exercise type is set to European (if set to Bermudan, and only one exercise date is present, this does *not* count as European!)
- the underlying has exactly two legs, one payer, one receiever and one of type "Fixed", one of type "Floating"
- the underlying notional is constant
- the underlying fixed rate is constant
- the underlying float margin is constant
- the underlying float gearing is constant
- the underlying float index is Ibor (OIS, BMA do not count as Standard) and fixes in advance (i.e. not in arrears)
- the underlying has no embedded caps / floors

If a trade is not classified as "standard European", it is priced using the Bermudan pricer, see 5.1.18.

Black model - log-normal volatilities

A swaption can be regarded as a single option on the forward swap rate with repeated payoffs. Assuming the forward swap rate $F(t; t_s, t_e)$ for swaption expiry (start) t_s, to underlying swap maturity (end) t_e follows Geometric Brownian Motion under the swap forward measure,

$$dF/F = \sigma_{t_s,t_e}(t)\, dW(t)$$

Then the Black model gives the following analytical solution for the present value of a European swaption (assuming log-normal volatilities):

$$NPV = N \sum_{i=1}^{n} \text{Black}(K, F(t_0; t_s, t_e), \sigma_{t_s,t_e}\sqrt{t_s}, \omega) \cdot A_{t_s t_e}$$

where:

- N: notional
- K: strike rate
- $F(t_0, t_s, t_e)$: the forward projected swap rate from swaption expiry to underlying swap maturity at time 0
- σ_{t_s,t_e}: the volatility of the forward swap rate corresponding to the start t_s and end t_e of the period
- ω: 1 for a payer swaption, -1 for a receiver swaption
- $A_{t_s t_e}$: an annuity at time 0, for physical settlement and cash settlement using the collateralized cash price method, this is

  $$A_{t_s,t_e} = \sum_{i=s+1}^{e} \delta_i \, P(t_i)$$

  and for cash settlement using the par yield curve method, this is

  $$A_{t_s,t_e} = P(t_s) \sum_{i=s+1}^{e} \delta_i \, (1+y)^{-(t_i - t_s)}$$

  with y denoting the fair swap rate observed at the option expiry date.
- δ_i: the daycount fraction in years for the period from t_{i-1} to t_i
- $P(t_i)$: the discount factor for time t_i

See Black Model, Section 6.2, for more details.

Black model - shifted log-normal volatilities

For shifted log-normal volatilities, the present value of a European swaption uses the same analytical solution as above, but with forward swap rate F and strike K amended to include a displacement parameter. See Section 6.2.

Bachelier model - normal volatilities

Under the Bachelier model it is assumed the forward swap rate $F(t; t_s, t_e)$ for swaption expiry (start) t_s to underlying swap maturity (end) t_e follows Arithmetic Brownian Motion under the swap forward measure:

$$dF = \sigma_{t_s,t_e}(t) \, dW(t)$$

Then, the Bachelier model gives the following analytical solution for the present value of a European swaption, assuming normal volatilities:

$$NPV = N \sum_{i=1}^{n} \text{Bachelier}(K, F(t_0; t_s, t_e), \sigma_{t_s,t_e}\sqrt{t_s}, \omega) \cdot A_{t_s t_e}$$

See Bachelier Model, Section 6.1, for more details.

Numerical engines

Alternatively, a European Swaption can be priced using one of the numerical pricers that are designed for Bermudan / American Swaptions. See section 5.1.18 for possible configurations of the numerical integration and finite-difference pricers, they work for product type "EuropeanSwaption" as well.

5.1.18 Bermudan/American Swaption and non-standard European Swaption

Bermudan/American Swaptions and non-standard European Swaptions (see 5.1.17 for the criteria to distinguish standard from non-standard swaptions) are priced using a one-factor Linear Gauss Markov (LGM) model, which is closely related to the Hull-White one-factor model, see section 6.5 for details.

General Model Parametrization

The reversion and volatility type is specified in the pricing engine configuration, as shown in listing 235. The allowed values are "HullWhite" and "Hagan" for both parameters "ReversionType" and "VolatilityType", and all combinations except ReversionType = Hagan, VolatilityType = HullWhite are allowed.

Listing 236 shows an alterative configuration using a finite-difference pricer instead of numerical integration, see below in the section on rollback implementation for details.

Listing 237 shows a configuration for American swaptions. The configuration has the same parameters as the one for Bermudan swaptions except for the additional parameter ExerciseTimeStepsPerYear which controls how many exercise dates are effective generated for pricing purposes. Usually this parameter is set to the same value as the TimeStepsPerYear parameter that controls the FD Grid, i.e. on each finite-difference time step an exercise is implemented (if it falls into the exercise period specified in the trade). Note: It is possible to use engine type Grid for American swaptions, but this is known to be slow in comparison because of the many time steps to process in the rollback.

If ReversionType is set to HullWhite, the model parameter H is derived from a piecewise constant mean reversion function $\kappa(t)$ via

$$H(t) = A \int_0^t e^{-\int_0^s \kappa(u)du} ds + B \tag{17}$$

see [42], formula (4.10a). Here A is a scaling factor and B is a shift parameter. If ReversionType is set to Hagan on the other hand, the model parameter H is derived from a piecewise constant function $h > 0$ via

Listing 235: Bermudan Swaption pricing configuration Grid Engine

```
<Product type="BermudanSwaption">
  <Model>LGM</Model>
  <ModelParameters>
    <Parameter name="Calibration">Bootstrap</Parameter>
    <Parameter name="CalibrationStrategy">CoterminalDealStrike</Parameter>
    <Parameter name="ReferenceCalibrationGrid">400,3M</Parameter>
    <Parameter name="Reversion">0.0</Parameter>
    <Parameter name="ReversionType">HullWhite</Parameter>
    <Parameter name="Volatility">0.01</Parameter>
    <Parameter name="VolatilityType">Hagan</Parameter>
    <Parameter name="ShiftHorizon">0.5</Parameter>
    <Parameter name="Tolerance">0.10</Parameter>
  </ModelParameters>
  <Engine>Grid</Engine>
  <EngineParameters>
    <Parameter name="sy">5.0</Parameter>
    <Parameter name="ny">30</Parameter>
    <Parameter name="sx">5.0</Parameter>
    <Parameter name="nx">30</Parameter>
  </EngineParameters>
</Product>
```

$$H(t) = A \int_0^t h(s)\, ds + B \tag{18}$$

which results in a piecewise linear H. If the VolatilityType is set to HullWhite, the model parameter α is derived from a piecewise constant volatility function $\sigma(t)$ via

$$\alpha(t) = \frac{\sigma(t)}{H'(t)} \tag{19}$$

see [42], formula (4.9b). The function $\sigma(t)$ is the volatility of the Hull White model in its classic form, i.e. this variant exactly replicates a Hull White model with piecewise constant volatility. If the VolatilityType is set to Hagan, the model parameter α is assumed to be piecewise constant.

We note that the formulas 17, 18 and 19 can be implemented using closed-form expressions thanks to the piecewise constant nature of the input functions, i.e. no numerical integration is required.

Model Parametrization for Bermudan/American Swaptions

For Bermudan/American swaption pricing the reversion parameter $\kappa(t)$ (ReversionType = HullWhite) resp. $h(t)$ (ReversionType = Hagan) is restricted to a constant value, which is assumed to be given as an external parameter "Reversion", see listing 235.

The parameter "Calibration" can be set to one of the values "None", "Bootstrap", "BestFit".

In case of "None" the model will not be calibrated before pricing, instead the volatility will be set to the value specified as "Volatility". The volatility function can be made

Listing 236: Bermudan Swaption pricing configuration FD Engine

```
<Product type="BermudanSwaption">
  <Model>LGM</Model>
  <ModelParameters>
    <Parameter name="Calibration">Bootstrap</Parameter>
    <Parameter name="CalibrationStrategy">CoterminalDealStrike</Parameter>
    <Parameter name="ReferenceCalibrationGrid">400,3M</Parameter>
    <Parameter name="Reversion">0.0</Parameter>
    <Parameter name="ReversionType">HullWhite</Parameter>
    <Parameter name="Volatility">0.01</Parameter>
    <Parameter name="VolatilityType">Hagan</Parameter>
    <Parameter name="ShiftHorizon">0.5</Parameter>
    <Parameter name="Tolerance">0.10</Parameter>
  </ModelParameters>
  <Engine>FD</Engine>
  <EngineParameters>
    <Parameter name="Scheme">Douglas</Parameter>
    <Parameter name="StateGridPoints">64</Parameter>
    <Parameter name="TimeStepsPerYear">24</Parameter>
    <Parameter name="MesherEpsilon">1E-4</Parameter>
  </EngineParameters>
</Product>
```

time dependent by specifying a comma separated list of n volatilities along with a list of $n-1$ times as an additional parameter "VolatilityTimes". This setting is useful when an external model parameterization should be used.

In case of "Bootstrap" the model volatility function will be calibrated to a set of coterminal swaptions. The (constant) value given as the paramter "Volatility" will be used as the initial guess for the calibration in this case. "VolatilityTimes" will be ignored and instead overwritten by a time grid corresponding to the option expiries of the coterminal swaption calibration instruments. See below how the coterminal swaptions are constructed.

In case of "BestFit" a constant model volatility will be calibrated to a set of coterminal swaptions that is constructed in the same way as in the case of "Bootstrap".

The scaling parameter A is set to 1 and the shift parameter B is set to $-H(T)$ with T denoting 0.5 times the remaining time to maturity of the trade to price.

Construction of coterminal calibration swaptions for Bootstrap

The set of calibration instruments is determined as follows:

The swaption expiries are chosen as the exercise dates of the trade to price. If a parameter "ReferenceCalibrationGrid" is given, at most one calibration instrument is kept per grid interval, in the example given in listing 235 this means that at most one swaption per 3M interval is kept. This can be both useful to stabilize the calibrated model volatility function (avoid osciallations if calibration instrument expiries lie too close to each other) and to speed up the calibration step.

For American swaptions the "ReferenceCalibrationGrid" parameter is mandatory and effectively one instrument per period is generated if the expiry falls into the exercise range of the trade.

Listing 237: American Swaption pricing configuration FD Engine

```
<Product type="AmericanSwaption">
  <Model>LGM</Model>
  <ModelParameters>
    <Parameter name="Calibration">Bootstrap</Parameter>
    <Parameter name="CalibrationStrategy">CoterminalDealStrike</Parameter>
    <Parameter name="ReferenceCalibrationGrid">400,3M</Parameter>
    <Parameter name="Reversion">0.0</Parameter>
    <Parameter name="ReversionType">HullWhite</Parameter>
    <Parameter name="Volatility">0.01</Parameter>
    <Parameter name="VolatilityType">Hagan</Parameter>
    <Parameter name="ShiftHorizon">0.5</Parameter>
    <Parameter name="Tolerance">0.10</Parameter>
    <Parameter name="ExerciseTimeStepsPerYear">24</Parameter>
  </ModelParameters>
  <Engine>FD</Engine>
  <EngineParameters>
    <Parameter name="Scheme">Douglas</Parameter>
    <Parameter name="StateGridPoints">64</Parameter>
    <Parameter name="TimeStepsPerYear">24</Parameter>
    <Parameter name="MesherEpsilon">1E-4</Parameter>
  </EngineParameters>
</Product>
```

The underlying maturity of the calibration swaptions is chosen to be identical to the trade maturity.

The strikes of the calibration swaptions are chosen dependent on the parameter "CalibrationStrategy" (see listing 235). This parameter can be set to "None" (if "Calibration" is set to "None", too) or to "CoterminalATM" or to "CoterminalDealStrike".

In case of "CoterminalATM" the calibration swaption strikes are set to their fair forward swap level.

In case of "CoterminalDealStrike" the calibration swaption strikes are set to the trade strike. More precisely, the strike K of the calibration swaption corresponding to an exercise date d of the trade to price is set to

$$K = r - s \tag{20}$$

where r is the rate of the first fixed rate coupon with accrual start date greater or equal to d and s is the first floating rate coupon spread with accrual start date greater or equal to d.

The calibration swaptions are always constructed following the conventions of the relevant swaption surface / cube. In particular, the index tenor of these swaptions is set to the market swaptions tenor.

Numerical implementation of calibration

Both the bootstrap and best fit calibrations are implemented using a Levenberg-Marquardt optimizer. The target function is defined as the RMSE of errors

of calibration swaptions and the single swaption error is taken to be the relative price error e

$$e = \frac{p - m}{m} \tag{21}$$

where p is the model price of the swaption and m is the market price of the swaption.

If the market value of a calibration swaption is below 10^{-20} we replace the strike of the calibration swaption with the forward ATM level of that swaption to stabilize the calibration. Furthermore, if the market value of the calibration swaption (possibly after the strike amendment just mentioned) is below 10^{-8} we use an absolute price error

$$e = p - m \tag{22}$$

in the target function for this swaption instead of a relative one. Again, this is done to stabilize the calibration.

In the case of a best fit calibration we optimize the model parameters simultaneously. In case of a bootstrap calibration we perform a series of one dimensional optimizations for each calibration swaption in ascending order of their option expiry times.

The calibration is executed in terms of transformed parameters always, so that restrictions on positivity of parameters are ensured by the transformation, yet the optimization itself is run in terms of variables for which all real values are admissable.

After the optimizer has finished the value of the target function is compared against the parameter "Tolerance" in listing 235 and either the pricing is aborted or continued with an alert issued to the log depending on the global pricing parameter "ContinueOnCalibrationError".

We note that the discount curve used in the calibration step to price the calibrations swaption is the curve specified by the "IrLgmCalibration" configuration, usually the in-ccy OIS curve since the swaptions are collateralized in their trade currency.

Implementation of Roll Back

To compute conditional expectation E_t at time t

$$E_t\left(\frac{V(T)}{N(T)}\right) \tag{23}$$

of a deflated value $V(T)/N(T)$ with $T > t$, we use either numerical integration or a finite difference solver:

- For numerical integration we use the numerical integration scheme proposed in [41], section 4.1. The numerical parameters for this scheme is read from the parameters "sx", "sy", "nx", "ny", see listing 235.
- For the finite difference solver we use "StateGridPoints" points covering states that are attained with a probability of at least "MesherEpsilon" times 2. We use "TimeStepsPerYear" equidistant steps per year for the rollback in time direction. See listing 236, 237.

For each cashflow of the underlying of the swaption we associate the latest exercise date for which the cashflow is relevant. Here, relevant means that the cashflow is part of the exercise-into right at the exercise date. We then estimate an amount for the cashflow so that it is available on this latest relevant exercise date. The exact date on which the coupon amount is estimated is determined as follows:

- for FixedRate, Ibor, ON compounded, ON averaged, BMA coupons, the greater of the latest relevant exercise date and the evaluation date is taken
- for CappedFloored Ibor coupons the greater of the latest relevant exercise date, the evaluation date and the coupon fixing date is taken
- for CappedFloored ON compunded and ON averaged the greater of the latest relevant exercise date, the evaluatino date and the first of the relevant ON fixing dates of the coupon is taken

The coupon amount estimation is done using the LGM discount bond formula

$$P(t,T,x) = \frac{P(0,T)}{P(0,t)} e^{-(H(T)-H(t))x-0.5(H(T)^2-H(t)^2)\zeta(t)} \tag{24}$$

where x is the LGM model state, see [42], section 4.1, and $P(0,t)$ is the market zerxo bond with maturity t computed on the relevant forwarding curve of the coupon.

The amount of a coupon is estimated once during a rollback and then rolled back to prior exercise dates. The total set of dates used in the rollback comprises

- the simulation dates for all coupons as determined above
- the exercise dates of the swaption to price
- the valuation date itself

Since the computation of conditional expectations is exact up to numerical errors, no further intermediate time steps need to be added as it would for example be the case for a finite difference solver.

The numeraire to deflate the values during the rollback is calculated using the "Pricing" configuration discount curve, usually the cross-OIS curve associated to the VM colalteral currency of the trade to price.

5.1.19 Callable Swap

Callable Swaps are decomposed into a swap and a swaption, see 5.1.1 and 5.1.18 for details on the pricing methodology of both components.

5.1.20 CMS Spread Option, Capped/Floored CMS Spread

European CMS Spread Options (or Caps/Floors on CMS Spreads) are priced using the bivariate swap rate model BrigoMercurio, see section 6.4. This model supports (shifted) log-normal and normal swap rate dynamics.

5.1.21 Digital CMS Spread Option

Similar to CMS Spread Options, Digital CMS Spread Options are priced using a bivariate swap rate model, see section 6.4, while the digital coupon payoffs in the underlying CMS swap are replicated using the call/put option spreads.

5.1.22 Flexi Swap

Flexi Swaps are priced following the Replication Approach described in (F. Jamshidian, 2005). The replicating basket of Bermudan Swaptions is priced using the method as described under Product Type "Bermudan Swaption", but using one global calibration for all swaptions derived from the flexi swap structure.

Prepayment option types "ReductionByAbsoluteAmount" and "ReductionUpToAbsoluteAmount" are both treated approximately by generating a schedule of deterministic, lower notional bounds from them assuming that all earlier prepayment options were executed.

5.1.23 Balance Guaranteed Swap

BGS are priced using an auxiliary Flexi Swap as a proxy. The amortization schedule of the Flexi Swap is set up as the notional schedule of the BGS assuming a zero CPR (Conditional Prepayment Rate). The lower notional bound of the Flexi Swap is constructed assuming a MaxCPR (Maximum Conditional Prepayment Rate) which is dependent on the Reference Security. The MaxCPR is estimated on the basis of the current CPR, historical CPRs and / or expert judgement as to provide a (hypothetical) sufficiently realistic hedge for the BGS. The option holder in the Flexi Swap is the payer of the structured leg (i.e. the leg replicating the payments of the reference security) in the BGS.

5.1.24 Interest Rate Swap with Formula Based Coupon

The present value of an Interest rate Swap leg with formula-based coupons can be written as follows

$$NPV_{form} = \sum_{i=1}^{n} N_i \, c_i^{form} \, \delta_i \, P(t_i)$$

where each formula based coupon is evaluated as an expectation in the T-forward measure associated with the payment time $T = t_i$

$$c_i^{form} = \mathbb{E}^{t_i} \left[f_i(R_1, \ldots, R_m) \right]$$

and $f_i(\cdot)$ is a function of IBOR and/or CMS rates as described in section 2.3.35.

Pricing Model

For non-callable structured coupons, we generalise the model in [31], 13.16.2. We assume the rates to evolve with a shifted lognormal dynamics under the T-forward measure in the respective currency of R_i as

$$dR_i = \mu_i(R_i + d_i)dt + \sigma_i(R_i + d_i)dZ_i \tag{25}$$

with displacements $d_i > 0$, drifts μ_i and volatilities σ_i or alternatively with normal dynamics

$$dR_i = \mu_i dt + \sigma_i dZ_i \tag{26}$$

for $i = 1, \ldots, m$ where in both cases Z_i are correlated Brownian motions

$$dZ_i dZ_j = \rho_{i,j} dt$$

with a positive semi-definite correlation matrix $(\rho_{i,j})$. The drifts μ_i are determined using given convexity adjustments

$$c_i = E^T(R_i(t)) - R_i(0)$$

where the expectation is taken in the T-forward measure in the currency of the respective rate R_i. Slightly abusing notation c_i can be computed both using a model consistent with 25 resp. 26 (i.e. a Black76 style model) or also a different model like e.g. a full smile TSR model to compute CMS adjustments. While the latter choice formally introduces a model inconsistency it might still produce more market consistent prices at the end of the day, since it centers the distributions of the R_i around a mean that better captures the market implied convexity adjustments of the rates R_i entering the structured coupon payoff.

Under shifted lognormal dynamics the average drift is then given by

$$\mu_i = \frac{\log((R_i(0) + d_i + c_i)/(R_i(0) + d_i))}{t}$$

and under normal dynamics

$$\mu_i = c_i/t$$

The NPV ν of the coupon is now given by

$$\nu = P(0,T)\tau E^T(f(R_1(t), \ldots, R_n(t))) \tag{27}$$

where $P(0,T)$ is the applicable discount factor for the payment time in the domestic currency and the expectation is taken in the T-forward measure in the domestic currency. To adjust 25 resp. 26 for the measure change between the currency of R_i and the domestic currency (if applicable), we apply the usual Black quanto adjustment

$$\mu_i \to \mu_i + \sigma_i \sigma_{i,X} \rho_{i,X}$$

where $\sigma_{i,X}$ is the volatility of the applicable FX rate and $\rho_{i,X}$ is the correlation between the Brownian motion driving the FX process and Z_i.

Evaluation

To evaluate the expectation in 27 we use Monte Carlo simulation to generate samples of the distribution of $(R_1, \ldots, R_n)$, evaluate f on these samples and average the results.

Using Monte Carlo simulation for each coupon has an impact on pricing performance (speed). The formula based coupon should therefore only be used in situations where explicit analytical or semi-analytical formulas are not available.

5.2 Foreign Exchange Derivatives

Vanilla FX instruments are priced using a multi-currency framework of discount curves by cash flow currency and forwarding curves per index, see 7.

5.2.1 FX Forward

Vanilla FX instruments are priced using a multi-currency framework of discount curves by cash flow currency and forwarding curves per index, see 7.

The NPV in base currency, for an FX Forward paying currency B and receiving currency A at time T, is computed as follows:

$$NPV = N_A \cdot X_A \cdot P_A(T) - N_B \cdot X_B \cdot P_B(T)$$

Where:

- N_A: the set amount to be received in currency A at maturity
- $P_A(T)$: the discount factor at maturity T for currency A
- N_B: the set amount to be received in currency B at maturity
- $P_B(T)$: the discount factor at maturity T for currency B
- X_A: the FX spot rate between currencies A and base currency
- X_B: the FX spot rate between currencies B and base currency

Note that for the case that USD is VM Collateral Currency, discounting is done using the USD OIS curve. All currencies other than the VM Collateral Currency use a "Foreign Currency Discount Curve" as the discount curve as described in section 7.

5.2.2 FX Swap

The FX Swap is priced as a portfolio of two FX Forwards (see 5.2.1), where the "near" leg is disregarded if the cash flow time has passed (and the NPV vanishes when the the "far" leg's cash flow time has passed as well).

5.2.3 European FX Option

Vanilla European FX Options are priced analytically using the Garman-Kohlhagen model and log-normal volatilities. Deterministic IR rates are assumed. Volatility structures with smile are used where available.

The Garman-Kohlhagen model assumes that the spot FX rate X follows Geometric Brownian Motion:

$$dX/X = \mu(t)\,dt + \sigma(t)\,dW$$

The model?s drift is calibrated such that the expected future spot rate agrees with today?s fair FX forward rate for exchange at time T. This means that the FX forward rate $F(t,T)$ follows drift free GBM under the T-forward measure:

$$dF(t,T)/F(t,T) = \sigma(t)\,F(t,T)\,dW$$

This leads to a Black76 analytical solution for the present value of a European FX option:

$$NPV = N \cdot \text{Black}(K, F(0,T), \sigma\sqrt{T}, \omega) \cdot P(T)$$

where:

- N: notional
- K: strike FX rate
- $F(0,T)$: the forward projected FX rate for maturity T at time 0 (valuation date)
- σ: the volatility of the FX forward rate
- ω: 1 for a call option (ie receiving spot FX and paying strike), -1 for a put option
- $P(T)$: the discount factor for maturity time T

See Black Model, Section 6.2, for more details.

5.2.4 American FX Option

Vanilla American FX Options can be either priced

- approximately using the Barone-Adesi and Whaley model with time-independent deal strike volatility. See Section 6.6, or
- numerically using Finite Difference Methods with the Black-Scholes framework and time-dependent deal strike volatility. Parameters are for setting the grid size and damping steps to limit the expansion of the grid. For discretization in time, number of grid points is scaled by bucketed expiry of the deal. For discretization in FX spot, there is no scaling. Damping steps, where Implicit Euler Scheme is used, are added to the beginning of discretization in time when the model rolls back from option maturity.

  Note that the current implementation assumes payoff at exercise.

5.2.5 FX Barrier Option

FX Barrier Options are priced analytically using formulas developed by [47] and [50] based on the Black-Scholes framework, see [43] Ch 4.17.1.

5.2.6 FX European Barrier Option

In a FX European Barrier Option both the barrier and the underlying option are European and have the same expiry date. The NPV can therefore be directly computed by replication of the payoff at expiry. The replication strategy makes use of the following four options

1. European FX Option Call(K) or Put(K) with strike K.
2. European FX Option Call(B) or Put(B) with strike B.
3. Digital FX Option DigiCall(K, abs(B-K)) with strike K and payout $abs(B-K)$.
4. Digital FX Option DigiCall(K, R) with strike K and payout R.

where K is the strike, B is the barrier and R is the rebate of the FX European Barrier Option.

The rebate is a Digital Option DigiCall(K, R) if barrier = UpOut or DownIn and a DigiPut(K, R) if barrier = UpIn or DownOut. I.e. the rebate amount is paid when the underlying option cannot be exercised due to the barrier conditions at expiry.

The barrier option itself is replicated as in table 31

Option Type	Barrier Type	Case	NPV
Call	UpIn or DownOut	$B > K$	Call(B) + DigiCall(B, B-K)
Call	UpIn or DownOut	$B < K$	Call(K)
Call	UpOut or DownIn	$B > K$	Call(K) - Call(B) -DigiCall(B, B-K)
Call	UpOut or DownIn	$B < K$	0
Put	UpIn or DownOut	$B > K$	0
Put	UpIn or DownOut	$B < K$	Put(K) – Put(B) - DigiPut(B, K-B)
Put	UpOut or DownIn	$B > K$	Put(K)
Put	UpOut or DownIn	$B < K$	Put(B) + DigiPut(B, K-B)

Table 31: FX European Barrier Option Replication

5.2.7 FX KIKO Barrier Option

FX KIKO Barrier Options are priced by replication, this replication strategy is briefly described in [53] and [54] The strategy makes use of FX Barrier Options and FX Double Barrier Options. There are three replications cases:

1. **The Knock-In barrier has been triggered before the evaluation date**: A knock-Out option with the knock-out barrier level is priced.
2. **The knock-In barrier is untriggered, the FX spot is between the two barriers**: This is replicated as follow

   $$NPV(barrier_{knockOut}, barrier_{knockIn}) = NPV_{KO}(barrier_{knockOut}) - NPV_{DKO}(barrier_{knockOut}, barr$$

   where NPV_{KO} is the NPV for a knock-out single barrier option, and NPV_{DKO} is the NPV for a double barrier knock-out option.

3. **The knock-In barrier is untriggered, the FX spot is above or below the two barriers**: This is replicated as follows:

$$NPV(barrier_{knockOut}, barrier_{knockIn}) = NPV_{KO}(barrier_{knockOut}) - NPV_{KO}(barrier_{knockIn})$$

5.2.8 Digital FX Option

Digital FX Options are cash or nothing instruments on an underlying FX rate. They can be either priced

- analytically using formulas developed by [50] based on the Black-Scholes framework, see [43] Ch 4.19.2, or
- as call respectively put spreads

In the current implementation, digital FX Options are priced analytically as described in section 5.2.8.

Analytical Digital FX Option

The NPV of a Digital FX option can be expressed as:

$$NPV = N \cdot P(T) \cdot \Phi(\omega\, d), \qquad d = \frac{1}{\sigma\sqrt{T}}\left(\ln\frac{F_T}{K} - \frac{\sigma^2 T}{2}\right)$$

where:

- N: the amount to be paid at expiration if the option is in-the-money
- $P(T)$: the discount factor for option expiry time T
- $\Phi(\cdot)$: the cumulative standard normal distribution
- ω: 1 for a call option, -1 for a put option
- F_T: the forward projected FX rate for time T at time 0
- K: the strike FX rate
- σ: the volatility FX forward rate

Digital FX Option as Call/Put Spread

Alternatively, when a more accurate reflection of volatility smiles is a priority, Digital FX options can be priced using call/put spreads (i.e. by approximating the digital FX option payment with the difference between two European vanilla FX options). Given a digital call/put option with strike rate K, the two v anilla call/put options will have strikes $K_- \leq K$ and $K_+ \geq K$, where $K_\pm = K \pm \epsilon$ and ϵ is small.

The present value of digital FX call and put options with strike K and payout amount M in currency A, can be expressed in terms of bought and sold vanilla FX calls/puts to exchange amount N of currency B for amount $K_\pm \cdot N$ in currency A. All PV?s are expressed in currency A:

$$NPV_{DigitalCall}(M, K) \approx NPV_{Call}(N, K_-) - NPV_{Call}(N, K_+)$$
$$NPV_{DigitalPut}(M, K) \approx NPV_{Put}(N, K_+) - NPV_{Put}(N, K_-)$$

where:

- M: the amount in currency A to be paid at expiration if the digital call/put is in-the-money
- $N = M/(K_+ - K_-)$

The European vanilla FX options are priced with the Garman-Kohlhagen / Black76 analytic formula (Section 5.2.3).

5.2.9 Digital FX Barrier Option

Digital FX Barrier Options are priced using Finite Difference Methods with the Black-Scholes framework. Parameters are for setting the grid size and damping steps to limit the expansion of the grid.

5.2.10 FX Touch Option

FX Touch Options are priced analytically using the Garman-Kohlhagen model and log-normal volatilities. Deterministic IR rates are assumed. Volatility structures with smile are used where available.

The Garman-Kohlhagen model assumes that the spot FX rate X follows Geometric Brownian Motion:

$$dX/X = \mu(t)\,dt + \sigma(t)\,dW$$

5.2.11 FX Double Barrier Option

FX Double Barrier Options are priced analytically using formulas developed by [47] and [50] based on the Black-Scholes framework, see [43] Ch 4.17.3.

5.2.12 FX Double Touch Option

FX Double Touch Options are priced analytically using formulas developed by [47] and [50] based on the Black-Scholes framework, see [43] Ch 4.19.6.

5.2.13 FX Asian Option

FX Asian Option is a simplified wrapper around the more general Basket option class which is in the scripted trade framework. The pricing of such a product is given by the Monte Carlo simulation.

5.2.14 FX Variance Swap

Commodity Variance Swaps and volatility swaps are priced using a replicating portfolio method [66]. See section 5.4.12 for the equivalent pricing of an Equity Variance Swap.

The pricing engine set up is, in analogy to Equity Variance Swaps:

5.3 Inflation Derivatives

5.3.1 CPI Swap, Zero Coupon Inflation Index Swap

A CPI Swap uses the inflation fixing at the start of the contract as base for all flows. Thus, the flows can be priced model-independently by discounting off the zero-coupon

Listing 238: "Robust" FX Variance Swap pricing engine configuration.

```
<Product type="FxVarianceSwap">
  <Model>BlackScholesMerton</Model>
  <ModelParameters/>
  <Engine>ReplicatingVarianceSwapEngine</Engine>
  <EngineParameters>
    <Parameter name="Scheme">Segment</Parameter>
    <Parameter name="Bounds">PriceThreshold</Parameter>
    <Parameter name="Steps">1000</Parameter>
    <Parameter name="PriceThreshold">1E-10</Parameter>
    <Parameter name="MaxPriceThresholdSteps">500</Parameter>
    <Parameter name="PriceThresholdStep">0.1</Parameter>
  </EngineParameters>
</Product>
```

inflation forward curve.

The present value of a CPI Swap leg with a final notional exchange can be expressed as follows:

$$NPV_{CPILeg} = N \cdot \sum_{i=1}^{n} r \frac{I(t_i)}{I(t_0)} \cdot \delta_i \cdot P(t_i) + N \cdot \frac{I(t_n)}{I(t_0)} \cdot P(t_n)$$

where:

- N: notional
- r: the contractual real rate
- $I(t_i)$: the relevant forward CPI fixing for time t_i
- $I(t_0)$: the relevant CPI fixing before issue date (base date)
- $I(t_n)$: the relevant forward CPI fixing for the maturity date
- δ_i: the daycount fraction for the accrual period from t_{i-1} up to time t_i
- $P(t_i)$: the discount factor for time t_i
- $P(t_n)$: the discount factor for maturity t_n

Alternatively, the term sheet may specify to subtract the un-inflated notional from each coupon payment.

$$NPV_{CPILeg} = N \cdot \sum_{i=1}^{n} r \left(\frac{I(t_i)}{I(t_0)} - 1 \right) \cdot \delta_i \cdot P(t_i) + N \cdot \frac{I(t_n)}{I(t_0)} \cdot P(t_n)$$

Furthermore, the term sheet may specify to subtract the un-inflated redemption, depending on whether the second floating leg of the CPI Swap contains a final notional exchange:

$$NPV_{CPILeg} = N \cdot \sum_{i=1}^{n} r \frac{I(t_i)}{I(t_0)} \cdot \delta_i \cdot P(t_i) + N \cdot \left(\frac{I(t_n)}{I(t_0)} - 1 \right) \cdot P(t_n)$$

5.3.2 CPI Cap/Floor

The CPI Cap/Floor payoff is

$$\Pi_{ZCIICF}(T) = N\left(\omega\left(\frac{I(T)}{I(T_0)} - K\right)\right)^+$$

where $\omega = 1$ for the Cap and $\omega = -1$ for the Floor.

To price the Cap/Floor, one considers the *Forward Inflation Index*

$$F(t,T) = \frac{I(t)}{I(T_0)}\frac{P_r(t,T)}{P(t,T)}, \qquad F(T,T) = \frac{I(T)}{I(T_0)}$$

where $P_r(t,T)$ is the "Real Rate Bond Price" curve stripped from CPI Swap quotes. $F(t,T)$ is a ratio of tradable assets and a Martingale in the T-Forward measure.

We then assume that $F(t,T)$ follows Geometric Brownian Motion without drift in the T-Forward measure, so that we obtain a Black pricing formula for the CPI Cap/Floor

$$NPV(0) = N\,P(T)\,\text{Black}(K, F(0,T), \sigma\sqrt{T}, \omega)$$

CPI Caps/Floors are primarily quoted in terms of cap and floor prices by maturity and strike rate r (with $K = (1+r)^{T-T_0}$). We convert them to equivalent Black volatilities $\sigma_{r,T}$.

5.3.3 Year-on-Year Inflation Swap

A Year-on-Year Inflation Swap is priced by discounting cashflows based on the Year-on-Year Inflation Index forwarding curve:

$$NPV_{YoYLeg} = N \cdot \sum_{i=1}^{n}\left(\frac{I(t_i)}{I(t_{i-1})} - 1\right)\cdot\delta_i \cdot P(t_i)$$

where $I(t_i)$ is the relevant forward Year-on-Year inflation fixing for time t_i.

For CPI and Year-on-Year inflation indices, caps and floors are priced using the Black model (see Section 7.4.2) for log-normal and shifted log-normal volatilities, and the Bachelier model (see Section 7.4.1) for normal volatilities. Pricing is analogous to pricing of caps and floors for IBOR indices.

5.3.4 Year-on-Year Cap/Floor

The Year-on-Year Inflation Indexed Swap exchanges a series of "Swaplet" payoffs

$$\Pi_{YYIIS}(T) = y(T) = \frac{I(T)}{I(S)} - 1$$

for fixed payments, each paid at respective period end T.
A related Caplet/Floorlet payoff, fixed at period end T, is:

$$\Pi_{YYIICF}(T) = (\omega(y(T) - K))^+$$

The time t price of the Caplet/Floorlet (in the T-Forward measure) is then

$$NPV_{YYIICF}(t) = P(t,T)\ \mathbb{E}_t^T\left[(\omega(y(T)-K))^+\right]$$

Consider the *Forward Year-on-Year Rate* as seen at time t

$$Y(t) = \frac{\Pi_{YYIIS}(t)}{P(t,T)} = \mathbb{E}_t^T\left[y(T)\right], \qquad Y(T) = y(T),$$

a ratio of tradable assets and a Martingale in the T-Forward measure.

Assuming e.g. $dY(t) = \sigma\, dW(t)$ then leads to a Bachelier formula for the year-on-year caplet/floorlet price as of today

$$NPV(0) = P(t,T)\ \text{Bachelier}(K, Y(t), \sigma\sqrt{T}, \omega)$$

Caps/Floors are composed of a series of YYII Caplets/Floorlets. YYII Caps/Floors are quoted in terms of prices by maturity and strike in a range from negative to positive, e.g. -1% to 6%. We convert them to equivalent normal volatilities $\sigma_{r,T}$.

5.4 Equity Derivatives

5.4.1 Equity Forward

The equity forwarding curve is used for respective underlying equity. The net present value of an equity forward can be priced discounting the cashflows at maturity as:

$$NPV = \text{Quantity} \cdot \omega(S(0,T) - K) \cdot P(T)$$

where:

- Quantity : number of shares of the underlying equity
- K: Strike price
- $S(0,T)$: the forward equity price for maturity T at valuation time 0 (calibrated to incorporate dividends, (Lichters, Stamm, & Gallagher, 2015) Ch 14.1
- $P(t)$: the discount factor for maturity time T
- ω: 1 for a forward to buy equity, -1 for a forward to sell equity

5.4.2 Equity Swap

An Equity Leg is priced by discounting cashflows based on the equity forwarding curve:

$$NPV_{EqLeg} = \sum_{i=1}^{n} N(t_{i-1}) \cdot \left(\frac{(s(t_i) + \delta(t_i))\text{FX}_i - s(t_{i-1}) \cdot \text{FX}_{i-1}}{s(t_{i-1}) \cdot \text{FX}_{i-1}} \right) \cdot P(t_{p,i})$$

where:

- $t_0, \ldots, t_n$ the times defining the equity fixing schedule
- $t_{p,1}, \ldots, t_{p,n}$ the times defining the coupon payment schedule

- $N(t_i)$: the notional at time t_i. If the notional is non-resettable this is given as a fixed number in the terms of the swap. If the notional is resettable, we have

$$N(t_i) = \frac{N(t_0)}{s(t_0)\mathrm{FX}_0} s(t_i)\mathrm{FX}_i = q s(t_i)\mathrm{FX}_i$$

  with a quantity $q := N(t_0)/[s(t_0)\mathrm{FX}_0]$ that is fixed for all coupons.

- $s(t_i)$: the equity forward price at time t_i
- $\delta(t_i)$: the dividend between time t_{i-1} and t_i for Total Return Swap; 0 for Price Return Swap
- $P(t_{p,i})$: the discount factor for time $t_{p,i}$
- FX_i: the fx rate converting the equity ccy to the equity leg ccy (if FX conversion is applicable, otherwise set this to 1 in the above formula)

5.4.3 European Equity Option

Vanilla European Equity Options are priced analytically using the Black-Scholes model and log-normal volatilities. The equity forwarding curve is used for respective underlying equity.

The Black-Scholes model assumes that the spot price of the underlying equity S follows Geometric Brownian Motion:

$$dS/S = \mu(t)dt + \sigma(t)\, dW$$

The model is calibrated such that the expected future equity spot price agrees with today?s fair equity forward price for time T. This means that the equity forward price $S(t,T)$ follows drift-free GBM under the T-forward measure:

$$dS(t,T)/S(t,T) = \sigma(t)\, dW$$

This leads a Black76 analytical solution for the present value of a European equity option:

$$NPV = \text{Quantity} \cdot \text{Black}(K, S(0,T), \sigma\sqrt{T}, \omega) \cdot P(T)$$

where:

- Quantity: number of shares of the underlying equity
- K: strike price
- $S(0,T)$: the forward equity price for maturity T at time 0 (valuation date)
- σ: the volatility of the equity forward price
- ω: 1 for a call option, -1 for a put option
- $P(T)$: the discount factor for maturity time T

See Black Model, Section 6.2.

5.4.4 American Equity Option

Vanilla American Equity Options are priced using the same model in American FX Option. See Section 5.2.4

5.4.5 Equity Barrier Option

Equity Barrier Options are priced analytically using formulas developed by [47] and [50] based on the Black-Scholes framework, see [43] Ch 4.17.1.

5.4.6 Equity European Barrier Option

Equity European Barrier Options are priced by replication. The strategy makes use of the following four options

1. European Equity Option with strike K.
2. European Equity Option with strike B.
3. Digital Equity Option with strike K and payout $abs(B - K)$.
4. Digital Equity Option with strike K and payout R.

where K is the strike, B is the barrier and R is the rebate of the Equity European Barrier Option.

5.4.7 European Equity Composite Option

Vanilla European Composite Options are priced analytically using the Black-Scholes model and log-normal volatilities.

The Black-Scholes model assumes that the spot price of the underlying equity S and the FX spot price X follow a Geometric Brownian Motion:

$$dS/S = \mu_S(t)dt + \sigma_S(t)\, dW$$
$$dX/X = \mu_X(t)dt + \sigma_X(t)\, dW_X$$

with correlated standard Brownian motions W and W_X, with correlation coefficient ρ.

The final payoff depends on the product of both processes $X(T)S(T)$.

The present value is given by Black Scholes Merton formula using the transformed volatility of the underlying into payoff currency

$$\sigma = \sqrt{\sigma_S^2 + \sigma_X^2 - 2\rho\sigma^S\sigma^X}$$

and the risk free rate of the payoff currency.

5.4.8 Equity Digital Option

Equity Digital Options are cash-or-nothing instruments on an underlying Equity spot rate. They can be either priced

- analytically using formulas developed by [50] based on the Black-Scholes framework, see [43] Ch 4.19.2, or
- as call respectively put spreads

In the current implementation, Equity Digital Options are priced analytically as described in section 5.4.8.

Analytical Equity Digital Option

The NPV of a Equity Digital Option can be expressed as:

$$NPV = \text{Quantity} \cdot N \cdot P(T) \cdot \Phi(\omega\, d), \qquad d = \frac{1}{\sigma\sqrt{T}}\left(\ln\frac{S(0,T)}{K} - \frac{\sigma^2 T}{2}\right)$$

where:

- Quantity: number of shares of the underlying equity
- N: the amount per number of shares to be paid at expiration if the option is in-the-money
- $P(T)$: the discount factor for option expiry time T
- $\Phi(\cdot)$: the cumulative standard normal distribution
- ω: 1 for a call option, -1 for a put option
- $S(0,T)$: the forward equity price for maturity T at time 0 (valuation date)
- K: strike price
- σ: the volatility of the equity forward price

Equity Digital Option as Call/Put Spread

Alternatively, when a more accurate reflection of volatility smiles is a priority, Equity Digital Options can be priced using call/put spreads (i.e. by approximating the Equity Digital Option payment with the difference between two European Vanilla Equity Options). Given a Digital Call/Put option with strike price K, the two V anilla Call/Put Options will have strikes $K_- \leq K$ and $K_+ \geq K$, where $K_\pm = K \pm \epsilon$ and ϵ is small.

The present value of one Equity Digital Call and Put Options with strike K and payout amount M in payoff currency A, can be expressed in terms of bought and sold Vanilla Equity Calls/Puts with underlying quantity Q and strike $K_\pm$ in currency A. All PVs are expressed in currency A:

$$NPV_{DigitalCall}(M,K) \approx NPV_{Call}(Q,K_-) - NPV_{Call}(Q,K_+)$$
$$NPV_{DigitalPut}(M,K) \approx NPV_{Put}(Q,K_+) - NPV_{Put}(Q,K_-)$$

where:

- M: the amount in payoff currency A to be paid at expiration if the Digital Call/Put is in-the-money
- $Q = M/(K_+ - K_-)$

The European Vanilla Equity Options are priced with the Black-Scholes / Black76 analytic formula (Section 5.4.3).

5.4.9 Equity Touch Option

Equity Touch Equity Options are priced analytically using the Black-Scholes model and log-normal volatilities. The equity forwarding curve is used for respective underlying equity.

The Black-Scholes model assumes that the spot price of the underlying equity S follows Geometric Brownian Motion:

$$dS/S = \mu(t)dt + \sigma(t)\,dW$$

5.4.10 Equity Double Touch Option

Equity Double Touch Options are priced analytically using formulas developed by [47] and [50] based on the Black-Scholes framework, see [43] Ch 4.19.6.

5.4.11 Equity Asian Option

Equity Asian Option is a simplified wrapper around the more general Basket option class which is in the scripted trade framework. The pricing of such a product is given by the Monte Carlo simulation.

5.4.12 Equity Variance and Volatility Swap

Equity Variance and volatility swaps are priced using a replicating portfolio method [66]. A variance swap can be theoretically replicated by a hedged portfolio of vanilla equity options on the same underlying with a range of strikes. The price of the variance swap involves the cost of the replicating portfolio as per below. The variance swap is then priced as

$$P(0,T) \cdot \omega \cdot N_{var} \cdot 10000 \cdot (Var - K_{var}) \tag{28}$$

A volatility swap on the other hand is priced as

$$P(0,T) \cdot \omega \cdot N_{vol} \cdot 100 \cdot (Vol - K_{vol}) \tag{29}$$

where:

- $\omega \in \{-1, 1\}$ is 1 for a long and -1 for a short position
- $K_{var} = K_{vol}^2$ is the variance strike which by definition is the square of the volatility strike K_{vol} given in the trade xml node "Strike"
- N_{var} is the variance notional which by convention is given by

  $$N_{var} = \frac{N_{vol}}{2 \cdot 100 \cdot K_{vol}}$$

  Here the vega notional N_{vol} is given in the trade xml node "Notional"
- Var is the annualized realized variance of RIC:.SPX at the end date of the trade
- $Vol = E(\sqrt{Var})$ is the expected volatility

- $P(0,T)$: the discount factor on the relevant OIS collateral curve for maturity time T

Accrued variance is handled by taking advantage of the additivity of the variance formula (after accounting for time). The future variance can be replicated (see [66], 4.5):

$$\text{Variance} = \frac{2}{T}\left[\int_0^F \frac{P(K)}{K^2}dK + \int_F^\infty \frac{C(K)}{K^2}dK\right] \tag{30}$$

where $P(K)$, $C(K)$ are non-discounted prices of puts resp. calls struck at strike K and F is the forward level of the underlying at T. The integrals are computed using a numerical integration scheme and parameters specified in the pricing engine configuration (see e.g. listing 239, 240) as follows:

- Scheme [Optional, default GaussLobatto]: GaussLobatto or Segment, this determines the integration scheme used
- Bounds [Optional, default PriceThreshold]: "Fixed": The integration bounds are found by

  $$\text{Lower} = Fe^{m\sigma\sqrt{T}}, \text{Upper} = Fe^{M\sigma\sqrt{T}}$$

  where m is the FixedMinStdDevs and M is the FixedMaxStdDevs parameter listed below and σ is the ATMF volatility of the underlying at maturity T.

  "PriceThreshold": The integration bounds are found by looking for the largest (smallest) strikes Lower, Upper such that the integrand in the replication formula above is below the PriceThreshold parameter listed below. The search starts at the forward level F for Lower, Upper and then Lower, Upper are updated by factors $1-p$ resp. $1+p$ where p is the PriceThresholdStep parameter below, until either the price threshold criterium is met or the number of updates exceeds the MaxPriceThresholdSteps parameter.
- Accuracy [Optional, default 10^{-5}]: Numerical accuracy tolerance for the numerical integration. This only applies to the GaussLobatto scheme.
- MaxIterations [Optional, default 1000]: The maximum number of iterations performed in the numerical integration. This only applies to the GaussLobatto scheme.
- Steps [Optional, default 100]: The number of steps in the Segment numerical integration scheme (only applies for this scheme).
- PriceThreshold [Optional, default 10^{-10}]: Used to determine the integration bounds if Bounds = PriceThredshold, see above.
- MaxPriceThresholdSteps [Optional, default 100]: Used to determine the integration bounds if Bounds = PriceThredshold, see above.
- PriceThresholdStep [Option, default 0.1]: Used to determine the integration bounds if Bounds = PriceThredshold, see above.

- FixedMinStdDevs [Optional, default -5]: Used to determine the integration bounds if Bounds = Fixed, see above.
- FixedMaxStdDevs [Optional, default 5]: Used to determine the integration bounds, if Bounds = Fixed, see above.
- StaticTodaysSpot [Optional, default false]: If true the contribution to the variance from the last day before the valuation date to the valuation date is ignored in scenario / sensitivity calculations. See below for more details.

If the market provides smooth and arbitrage free smiles, the default values listed above are the recommended settings for the pricer. If the market smiles are of lower quality, the following settings can be used instead:

- listing 239 using PriceThreshold making sure that the entirety of the relevant integration domain is captured. This method is still dependent on the option price function to be sufficiently fast decreasing though.
- listing 240 using fixed integration bounds. This method is independent of the shape of the option price function, but might miss out parts of the integration domain with relevant contribution.

Notice that these methods - while ensuring more robust calculation - are possibly slower and less accurate.

Listing 239: "Robust" variance swap pricing engine configuration using PriceThreshold if market smiles are of lower quality.

```
<Product type="EquityVarianceSwap">
  <Model>BlackScholesMerton</Model>
  <ModelParameters>
    <Parameter name="StaticTodaysSpot">false</Parameter>
  </ModelParameters>
  <Engine>ReplicatingVarianceSwapEngine</Engine>
  <EngineParameters>
    <Parameter name="Scheme">Segment</Parameter>
    <Parameter name="Bounds">PriceThreshold</Parameter>
    <Parameter name="Steps">1000</Parameter>
    <Parameter name="PriceThreshold">1E-10</Parameter>
    <Parameter name="MaxPriceThresholdSteps">500</Parameter>
    <Parameter name="PriceThresholdStep">0.1</Parameter>
  </EngineParameters>
</Product>
```

Note on volatility swap pricing: The volatility Swap is priced using the same replication approach as the Variance Swap, plugging the square root of the expected variance from the Variance Swap pricing as realized volatility into the Volatility Swap's payoff.

This approach ignores model convexity, i.e.

$$Vol = E(\sqrt{Var}) \neq \sqrt{E(Var)}$$

Listing 240: Alternative "Robust" variance swap pricing engine configuration using fixed integration bounds.

```
<Product type="EquityVarianceSwap">
  <Model>BlackScholesMerton</Model>
  <ModelParameters>
    <Parameter name="StaticTodaysSpot">false</Parameter>
  </ModelParameters>
  <Engine>ReplicatingVarianceSwapEngine</Engine>
  <EngineParameters>
    <Parameter name="Scheme">Segment</Parameter>
    <Parameter name="Bounds">Fixed</Parameter>
    <Parameter name="Steps">1000</Parameter>
    <Parameter name="FixedMinStdDevs">-5.0</Parameter>
    <Parameter name="FixedMaxStdDevs">5.0</Parameter>
  </EngineParameters>
</Product>
```

in general. This is accurate only when the model's volatility function is deterministic and time-dependent only. Ignoring convexity has the effect of overestimating the Volatility Swap price (Jensen's inequality). The replication aproach is model-independent and makes use of market call and put prices, i.e. market smile, directly which is generally not consistent with deterministic model volatility functions.

Potential extensions would involve using local or stochastic volatility models, such as Heston's, calibrated to the market smile and evaluating the Volatility Swap payoff using Monte Carlo methods or approximate analytical expressions, see [33].

Step by step example: In this section we detail the steps involved in the valuation and sensitivity calculation of a variance swap. Consider the trade in listing 241. We assume the valuation date to be 2023-03-13. We have

- $\omega = 1$ (long position)
- $K_{vol} = 0.32$ and$K_{var} = 0.32^2 = 0.1024$
- $N_{vol} = 10000$ and $N_{var} = \frac{10000}{2 \cdot 100 \cdot 0.32} = 156.25$

for our example trade.

Accrued variance: Since the start date 2023-03-01 lies before the valuation date 2023-03-13, part of the realized variance Var entering the payoff is already determined by past end-of-day prices. Table 32 shows these prices p_i, $i = 0, \ldots, 8$, the resulting log-return r_i, $i = 1, \ldots, 8$ for the dates d_i, $i = 0, \ldots, 8$ (see "Date" column), calculated as

$$r_i = \ln \frac{p_i + div_i}{p_{i-1}}$$

where div_i denotes a dividend with ex-div date falling on date d_i.

In general, dividend payments are taken into account if "AddPastDividends" is set to "true" in the trade xml. The squared log-returns r_i^2 are added up in the column "Accrued Daily Variance".

Listing 241: Example Variance Swap Trade Details

```
<Trade id="EQ_VarianceSwap">
  <TradeType>EquityVarianceSwap</TradeType>
  <Envelope>
    <CounterParty>JPM</CounterParty>
    <NettingSetId>192837273</NettingSetId>
  </Envelope>
  <EquityVarianceSwapData>
    <StartDate>2023-03-01</StartDate>
    <EndDate>2021-05-01</EndDate>
    <Currency>USD</Currency>
    <Underlying>
      <Type>Equity</Type>
      <Name>RIC:.SPX</Name>
    </Underlying>
    <LongShort>Long</LongShort>
    <Strike>0.32</Strike>
    <Notional>10000</Notional>
    <Calendar>US</Calendar>
    <MomentType>Variance</MomentType>
    <AddPastDividends>true</AddPastDividends>
  </EquityVarianceSwapData>
</Trade>
```

The last entry 0.00111676 is converted to an annualized variance by multiplying with $252/n$ where $n = 8$ is the number of past returns taken into account. The annualized accrued variance 0.0351779 is available in the additional results as "accruedVariance".

Contribution of today's spot to the accrued variance under scenarios: When calculating a scenario npv that involves a shift of the valuation date's equity spot price (e.g. when calculating the sensitivity to the equity spot price) we provide the option to freeze the contribution of the return calculated for the evaluation date to the value that was calculated under the base scenario excluding all shifts.

In other words when this option is enabled, a shift of the equity spot will not impact the calculated accrued variance. The option is enabled by setting "StaticTodaysSpot" to true.

Future variance: The future variance is calculated using the replication integral 30. Using the market data as of 2023-03-13 we have

- a maturity time $T = 0.134247$
- a RIC:.SPX forward level of $F = 3880.45$ at maturity T
- an at-the-money volatility $\sigma = 0.225506$

Using the pricing engine config 239 the lower and upper integration bounds are computed as

$$\text{Lower} = Fe^{m\sigma\sqrt{T}} = 2567.23, \text{Upper} = Fe^{M\sigma\sqrt{T}} = 5865.42$$

Date	EQ Price	Log-Return	Accrued Daily Variance
2023-03-01	3951.39		
2023-03-02	3981.35	0.00755303	5.70483e-05
2023-03-03	4045.64	0.0160176	0.000313613
2023-03-06	4048.42	0.000686973	0.000314085
2023-03-07	3986.37	-0.0154444	0.000552614
2023-03-08	3992.01	0.00141392	0.000554613
2023-03-09	3918.32	-0.0186323	0.000901776
2023-03-10	3861.59	-0.0145844	0.00111448
2023-03-13	3855.76	-0.00151036	0.00111676
annualized			0.0351779

Table 32: Accrued daily variance, the annualized variance is calculated as "accrued daily variance x 252 / number of days".

where $m = -5$, $M = 5$. Figure 1 shows the implied volatility smile of the underlying RIC:.SPX at the swap maturity and figure 2 shows the integrand from 30.

The integral 30 is computed following the pricing engine config, in our case using a a segment integral with 100 steps. This results in an estimate for the future variance of 0.062806.

The future variance is available in the additional results under "futureVariance".

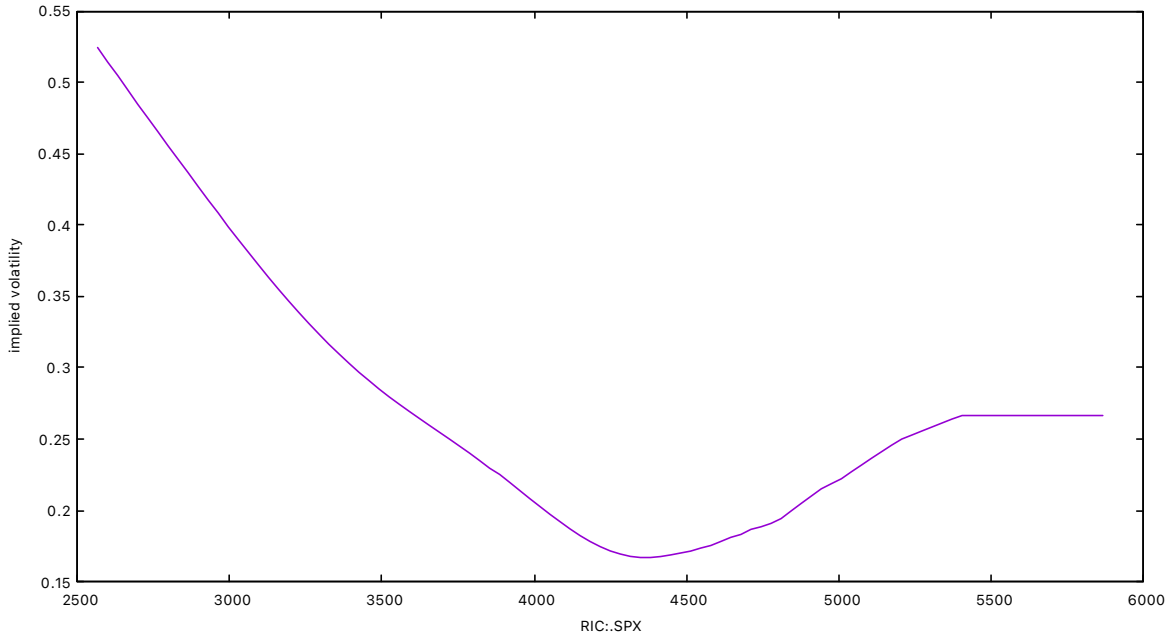

Figure 1: Implied volatility of RIC:.SPX at swap maturity $T = 0.134247$

Total variance: The total variance Var in 28 is computed as a weighted sum of the accrued and future variance

$$Var = \frac{n_1}{n}\text{accrued var} + \frac{n_2}{n}\text{future var}$$

where

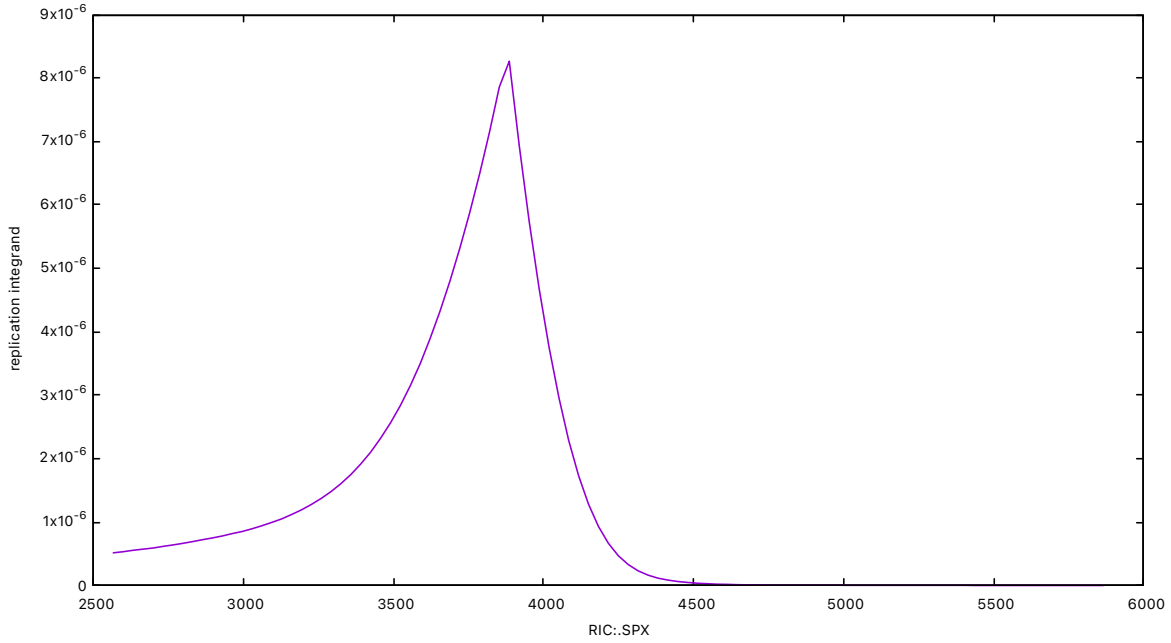

Figure 2: The replication integrand, which is given by $\frac{P(K)}{K^2}$ *for* $K < F$ *and* $\frac{C(K)}{K^2}$ *for* $K > F$

- n_1 is the number of business days between the swap start date and today (both included)
- n_2 is the number of business days between today (excluded) and swap maturity (included)
- n is the number of business days between the swap start and maturity date (both included)

The total variance is available in the additional results under "totalVariance". In our case we get

$$Var = \frac{9}{43} \cdot 0.035178 + \frac{34}{43} 0.062806 = 0.057023$$

Swap NPV: The NPV of the swap is finally computed by

$$P(0,T) \cdot \omega \cdot N_{var} \cdot 10000 \cdot (Var - K_{var})$$

with

- P(0,T) the relevat discount factor for maturity T, available in the additional results under "MaturityDiscountFactor", in our case 0.993666
- the long / short indicator ω which is 1 in our case
- the variance notional 156.25, available in the additional results under "VarianceNotional" (see above for the calculation)
- the variance strike 0.1024, available in the additional results under "VarianceStrike" (see above for the calculation)

This gives a total swap npv of $-70,452.47$ USD.

Sensitivity calculation: Sensitivities for a variance swap are calculated following a "bump and revalue" approach using the usual configured shift scenarios. The main sensitivities for a variance swap are the sensitivities to implied volatility (vega) and equity spot price shifts (delta).

There are two major factors impacting the sensitivity to the equity spot:

- Choice of smile dynamics: "Sticky Strike" and "Sticky Moneyness" yield different sensitivities to the equity spot. The latter choice makes the future variance insensitive to the spot shift, as can be seen from 30. The former will generate a significant sensitivity to the equity spot on the other hand. Both effects are intuitively clear as well.
- Inlcusion or Exclusion of today's shift contribution (configurable via pricing engine config "StaticTodaysSpot").

Table 33 illustrates this by summarizing the sensitivities of our example trade. Delta is computed w.r.t. a 1% relative shift of the equity spot price, vega w.r.t. to an absolute shift of 0.01 in the implied vol and rho w.r.t. a shift of 1 basispoint on interest rate curves.

Smile Dynamics	Todays Shift	Delta	Vega	Rho
Sticky Strike	Included	-3,622	7,357	-5
Sticky Strike	Excluded	-4,328	7,357	-5
Sticky Moneyness	Included	706	7,357	1
Sticky Moneyness	Excluded	0	7,357	1

Table 33: Variance Swap Sensitivities under different smile dynamics and inclusion / exclusion of today's shift contribution

5.4.13 Equity Position

The price of an equity position is directly derived from the market quote(s) of the underlying share(s). For a basket the weighting is taken into account and if equities in several currencies are present in the basket, the equity prices are converted to a common npv currency using the current FX Spot rates. The npv currency is by convention the currency of the first equity in the basket.

5.4.14 Equity Option Position

The price of an equity option position is derived from the prices of the underlying options, see 5.4.3 for details on how this is done. For a basket the weighting is taken into account and if equities in several currencies are present in the basket, the equity prices are converted to a common npv currency using the current FX Spot rates. The npv currency is by convention the currency of the first equity option in the basket.

5.4.15 Equity Outperformance Option

Equity Outperformace Options are priced analytically using formulas generalized from [31], 13.16.2.

5.5 FX / Equity / Commodity / Rates / Inflation Exotics

5.5.1 Double Digital Option

The double digital option is priced using correlated Black-Scholes processes for the two underlyings. Each process is calibrated to its quoted market ATM volatility. The correlation is an externally given parameter. The valuation is done using a Monte Carlo simulation.

See 5.5.33 for more details, the relevant product classification is MultiAssetOption.

5.5.2 Vanilla European Option Contingent on a Barrier

A European option with a barrier in another underlying is priced using correlated Black-Scholes processes for the two underlyings (option underlying and barrier underlying), unless the option underlying is an IR asset class, in which case we use Gaussian-Cam for the IR underlying. Each process is calibrated to its quoted market ATM volatility. The correlation is an externally given parameter. The valuation is done using a Monte Carlo simulation.

See 5.5.33 for more details, the relevant product classification is MultiAssetOption if both underlyings are Equity, FX or COMM. If the option underlying is IR, then the product classification is IrHybrid.

5.5.3 Autocallable Type 01

The autocallable option of type 01 is priced using a Black-Scholes process for the underlying which is calibrated to its quoted market ATM volatility. The valuation is done using a Monte Carlo simulation.

See 5.5.33 for more details, the relevant product classification is SingleAssetOption.

5.5.4 Performance Option Type 01

The performance option of type 01 is priced using correlated Black-Scholes processes for the underlyings. Each process is calibrated to the underlying's quoted market ATM volatility. The correlations are externally given parameters. The valuation is done using a Monte Carlo simulation.

See 5.5.33 for more details, the relevant product classification is MultiAssetOption.

5.5.5 Equity Cliquet Option

The cliquet option is priced using a Black-Scholes process for the underlying which is calibrated to its quoted market ATM volatility. The valuation is done using a Monte Carlo simulation.

5.5.6 Window Barrier Option

EQ/FX/COMM window barrier options are priced using a Black-Scholes process for the underlying calibrated to the option strike. The valuation is done using a Monte Carlo simulation.

See 5.5.33 for more details, the relevant product classification is SingleAssetOption.

5.5.7 Generic Barrier Option

Generic Barrier Options are priced using a finite-difference solver in a Black-Scholes model. The final payoff $V(T, S(T))$ at the option expiry T dependent on the underlying value $S(T)$ is easily generated, including an additional transatlantic barrier condition. We use the notation as in table 34

Variable	Meaning
$V_V(t, S)$	Value of the option conditional on (t, S), this excludes rebate payments and assumes barrier monitoring from t on
$V_{NA}(t, S)$	Value of the final payoff at t conditional on no knock events happening for times s with $s \geq t$
$V_{KI}(t, S)$	Value of the final payoff at t conditional on a KI event but no KO event happening for a time s with $s \geq t$
$V_{KO}(t, S)$	Value of the final payoff at t conditional on a KO event but no KI event happening for a time s with $s \geq t$
$V_{KIKO}(t, S)$	Value of the final payoff at t conditional on a KI event at time u and a KO event at v with $t \leq u < v$)
$V_{KOKI}(t, S)$	Value of the final payoff at t conditional on a KO event at time u and a KI event at v with $t \leq u < v$)

Table 34: Values in FD backward solver

The idea is that we track future Knock events. For a series of future knock events we collapse neighboring events of the same type "in" and "out" into a single event (because only the first is relevant). Also we only look two events into the future, since this is sufficient for the update rules below. Notice that for this purpose we ignore whether the events as allowed in this order according to the term sheet, i.e. we track a sequence of KI then KO events in V_{KIKO} even if KoBeforeKi is defined, i.e. the KO event is excluded by the termsheet in this case. Also notice, that V_x for $x \in \{NA, KI, KO, KIKO, KOKI\}$ is the value of the final payoff, i.e. without considering KI or KO events in the value itself. This is different for V_V, where we account for exluded barrier events and include the barrier events in the value.

At expiry we set $V_V = V$ if no KI barrier is defined and $V_V = 0$ if a KI barrier is defined. Furthermore we set $V_{NA} = V$ and $V_x = 0$ for $x \in \{KI, KO, KIKO, KOKI\}$. The rollback from t_{i+1} to t_i works as follows:

If there is no KI or KO event at t_i, all values are rolled back without any update.

If there is a KI event at t_i, we apply the following updates after the roll back:

- $V_{KIKO} := V_{KO} + V_{KIKO} + V_{KOKI}$
- $V_{KOKI} := 0$
- $V_{KI} := V_{NA} + V_{KI}$
- $V_{KO} := 0$
- $V_{NA} := 0$
- $V_V := V_{KI}$, add V_{KIKO} if KoBeforeKi

If there is a KO event at t_i, we apply the following updates after the roll back:

- $V_{KOKI} := V_{KI} + V_{KOKI} + V_{KIKO}$
- $V_{KIKO} := 0$
- $V_{KO} := V_{NA} + V_{KO}$
- $V_{KI} := 0$
- $V_{NA} := 0$
- $V_V := 0$ if KoAlways or KoBeforeKi

If the trade is seasoned we continue the rollback into the past to account for past KI and KO events properly. This rollback does not involve the computation of conditional expectations though, we only update the values according to the rules above.

Rebates are handled as follows:

The Transatlantic Barrier Rebate is incorporated into the final option payoff, i.e. for the nodes where the Transatlantic Barrier deactivates the option, a payoff amount equal to the rebate amount is generated.

If only KO-Barriers are present, the rebate NPV is computed in the PDE by setting V_V to the rebate amount (instead of 0) on nodes where a KO event is detected. The payment timing is accounted for by appropriate discounting of that amount.

If at least one KI-Barrier barrier is present, only a unique rebate amount is allowed which is always paid on the option settlement date. We can roll back this amount as R_x in complete analogy to the V_x for $x \in \{NA, KI, KO, KIKO, KOKI\}$. The variables R_x are initialised in the same way as V_x, but with the final option payoff replaced with the rebate payment R. The rebate NPV is then computed as the discounted value of R at $t = 0$ minus R_V (because R_V represents an option that pays R at maturity if the original barrier option is active).

When evaluated via Monte-Carlo simulation, the exercise probability and the trigger probability are given as additional results. Additional to this, the knock-out probability of the transatlantic feature is given as well if relevant. Therein, there are the following definitions active:

5.5.8 Best Entry Option

A Best Entry Option is priced in a Black-Scholes model, with the valuation done using a Monte Carlo simulation.

5.5.9 Basket Option

A basket option is a European option written on a basket of n underlying equities. The basket price is a weighted average of equity prices, taking FX conversions into the payoff currency into account.

Basket options are priced using correlated Black-Scholes processes for the underlyings S_i calibrated to their quoted market ATM volatility. The valuation is done using a Monte Carlo simulation.

See 5.5.33 for more details, the relevant product classification is MultiAssetOption.

Additional Result	Description
ExerciseProbability	The estimated probability of the exercise event. The latter means that the payoff is non-zero at maturity (i.e. the plain vanilla part of the payoff is positive and no knock-out event has accured and necessary knock-in events have occured until then).
TriggerProbability	The estimated probability of the exercise event. The latter means that the knock-in or the knock-out event (or one of them if there are multiple ones) has been triggered.
TransatlanticTriggered	The estimated probability of the transatlantic trigger event. The latter means that the transatlantic knock-out event (or at least one of them if there are multiple ones) has been triggerd.

Table 35: Additional results output labels for the Generic Barrier Option

5.5.10 Worst Of Basket Swap

A Worst Of Basket Swap is priced in a Hull White model calibrated to ATM coterminal swaptions. The valuation is done using a Monte Carlo simulation.

The final equity amount is determined by the strike price of the worst-performing asset at the final valuation date, taking FX conversions into the payoff currency into account.

See 5.5.33 for more details, the relevant product classification is MultiUnderlyingIrOption if the basket consists only of InterestRate underlyings. Otherwise (if the basket consists of Equity, FX or Commodity) the product classification is IrHybrid.

5.5.11 Rainbow Option

Rainbow options are Euopean calls or puts on the maximum or mimimum of a range of assets.

We price Rainbow options using correlated Black-Scholes processes for the underlyings calibrated to their quoted market ATM volatility. The valuation is done using a Monte Carlo simulation.

See 5.5.33 for more details, the relevant product classification is MultiAssetOption.

5.5.12 Exotic Variance and Volatility Derivatives

Exotic variance/volatility swaps and options are priced using a Black-Scholes process for the underlying (or underlyings for pairwise and basket variance swaps) calibrated to its quoted market ATM volatility. The valuation is done using a Monte Carlo simulation.

See 5.5.33 for more details, the relevant product classification is SingleAssetOption (or MultiAssetOption for pairwise and basket variance swaps).

5.5.13 Accumulators and Decumulators

EQ/FX accumulators and decumulators are priced using a Black-Scholes process for the underlying calibrated to its quoted market ATM volatility. The valuation is done using a Monte Carlo simulation.

See 5.5.33 for more details, the relevant product classification is SingleAssetOption.

5.5.14 Accumulators and Decumulators

5.5.15 Target Redemption Forward (TaRF)

An EQ/FX/Comm TaRF is priced using a Black-Scholes process for the underlying calibrated to its quoted market ATM volatility. The valuation is done using a Monte Carlo simulation.

See 5.5.33 for more details, the relevant product classification is SingleAssetOption.

5.5.16 Knock Out Swap

A Knock Out Swap is priced in a Hull White 1F model calibrated to ATM coterminal swaptions.

5.5.17 Window Barrier Options

Window Barrier Options are priced using a Black-Scholes process for the underlying calibrated to its quoted market ATM volatility. The valuation is done using a Monte Carlo simulation.

See 5.5.33 for more details, the relevant product classification is SingleAssetOption.

5.5.18 CMS / Libor Asian Cap Floor

Asian Cap / Floors on Libor / CMS underlyings are priced in a Hull White 1F model calibrated to ATM coterminal swaptions. The valuation is done using a Monte Carlo simulation.

See 5.5.33 for more details, the relevant product classification is SingleUnderlyingIrOption.

5.5.19 CMS Volatility Swap

Volatility Swap on CMS underlyings are priced in a Hull White 1F model calibrated to ATM coterminal swaptions. The valuation is done using a Monte Carlo simulation.

See 5.5.33 for more details, the relevant product classification is SingleUnderlyingIrOption.

5.5.20 Forward Volatility Agreement

A Forward Volatility Agreement is priced using a Black-Scholes process for the underlying calibrated to its quoted market ATM volatility. The premium of the underlying delta-neutral straddle option is calculated using the Black-76 model,

assuming a strike price equal to the forward rate. The valuation is done using a Monte Carlo simulation.

See 5.5.33 for more details, the relevant product classification is SingleAssetOption.

5.5.21 Correlation Swap

A Correlation Swap is priced using a Black-Scholes process for the underlyings calibrated to their quoted market ATM volatilities. The valuation is done using a Monte Carlo simulation.

See 5.5.33 for more details, the relevant product classification is MultiAssetOption.

5.5.22 Asset Linked Cliquet Option

An Asset Linked Cliquet Option is priced using a Hull White 1F model for the linked CMS underlying (calibrated to ATM coterminal swaptions) and a Black-Scholes process for all other underlyings.

5.5.23 CMS Cap / Floor with Barrier

A CMS Cap Floor with a barrier in an underlying index is priced using a Hull White 1F model for the CMS underlying/s (calibrated to ATM coterminal swaptions) and a Black-Scholes process for the barrier underlying (calibrated to their quoted market ATM volatilities). The valuation is done using a Monte Carlo simulation.

See 5.5.33 for more details, the relevant product classification is IrHybrid.

5.5.24 Fixed Strike Forward Starting Option

A Forward Starting Option with a fixed strike is priced using a Black-Scholes process for the underlying calibrated to its quoted market ATM volatility. The premium of the forward starting option is calculated using the Black-76 model. The valuation is done using a Monte Carlo simulation.

See 5.5.33 for more details, the relevant product classification is SingleAssetOption.

5.5.25 Floating Strike Forward Starting Option

A Forward Starting Option with a floating strike is priced using a Black-Scholes process for the underlying calibrated to its quoted market ATM volatility. The valuation is done using a Monte Carlo simulation.

See 5.5.33 for more details, the relevant product classification is SingleAssetOption.

5.5.26 Forward Starting Swaption

A Forward Starting Swaption is priced in a Hull White 1F model calibrated to ATM coterminal swaptions. The valuation is done using a Monte Carlo simulation.

See 5.5.33 for more details, the relevant product classification is SingleUnderlyingIrOption.

5.5.27 Ladder Lock-In Option

Ladder lock-in options are priced using a Black-Scholes process for the underlying calibrated to its quoted market ATM volatility. The valuation is done using a Monte Carlo simulation.

See 5.5.33 for more details, the relevant product classification is SingleAssetOption.

5.5.28 Floored Average CPI Zero Coupon Inflation Index Swap

A Floored Average CPI Zero Coupon Inflation Index Swap is priced in a Dodgson-Kainth (DK) model calibrated to coterminal swaptions and zero coupon inflation caps and floors. The valuation is done using a Monte Carlo simulation.

See 5.5.33 for more details, the relevant product classification is SingleUnderlyingIrOption.

5.5.29 Moving Maximum Year-on-Year Inflation Index Swap

A Moving Maximum Year-on-Year Inflation Index Swap is priced in a Dodgson-Kainth (DK) model calibrated to coterminal swaptions and zero coupon inflation caps and floors. The valuation is done using a Monte Carlo simulation.

See 5.5.33 for more details, the relevant product classification is SingleUnderlyingIrOption.

5.5.30 Irregular Year-on-Year Inflation Index Swap

A Irregular Year-on-Year Inflation Index Swapp is priced in a Dodgson-Kainth (DK) model calibrated to coterminal swaptions and zero coupon inflation caps and floors. The valuation is done using a Monte Carlo simulation.

See 5.5.33 for more details, the relevant product classification is SingleUnderlyingIrOption.

5.5.31 LPI Swap

A LPI Swap is priced in a Dodgson-Kainth (DK) model calibrated to coterminal swaptions and zero coupon inflation caps and floors. The valuation is done using a Monte Carlo simulation.

See 5.5.33 for more details, the relevant product classification is SingleUnderlyingIrOption.

5.5.32 Lapse Risk Hedge Swap

A Lapse Risk Hedge Swap is priced in a Hull White 1F model calibrated to ATM coterminal.

The percentage of redeemed units ΔN in the computation of the lapse risk hedge amount is assumed to be deterministic and the historical fixings and all forward projections are provided by the client in the trade xml. If the client does not provide any or all forward projections, we assume all missing future projections to be constant and equalt to the last known rate.

There are no price sensitivties in respect to ΔN computed and since there is volatility in the process of redeemed units the price for both embedded options are underestimated.

The valuation is done using a Monte Carlo simulation.

See 5.5.33 for more details, the relevant product classification is SingleUnderlyingIrOption.

5.5.33 Pricing Model Details for Scripted Trades

Depending on the product classification of a scripted trade we apply pricing models and parameters as listed in table 36. The meaning of these parameters are as follows:

- Model / Engine: The model and its numerical solution. The following pairs are available:
  - BlackScholes / MC: A single or multivariate Black Scholes process. For multivariate processes the Brownian motions are coupled by a time-constant instantaneous correlation. The process is evolved using exact moments, therefore in the single variate (or zero correlation) case large time steps between the relevant simulation times for a trade can be taken. For a multivariate process with non-zero correlations the given number of time steps per year is used to determine time intervals on which an average of the integrated volatility is used to compute the covariance with other process components. The process is evolved using Sobol low discrepancy sequences with Brownian Bridge interpolation. The volatility terms structure at the ATMF strike is used to evolve each process component.
  - GaussianCam / MC: IR components are represented by LGM1F models, FX, EQ, COMM components by Black Scholes processes with stochastic drift term and the Inf component by Dodgson-Kainth (DK) model. IR components are calibrated to ATM coterminal swaptions, the Inf component to coterminal inflation caps/floors, FX, EQ and COMM components to ATMF options. The calibration expiries are chosen as the relevant simulation times for each trade. If more than one component is present they are coupled by a time-constant instantaneous correlation. The process is evolved using exact moments, therefore large time steps between the relevant simulation times can be taken.
- Samples: The number of samples used in MC engines.
- Regression Order: If a product requires the computation of conditional expected NPVs (typically to take exercise decisions in Bermudan / American structures), the order of the regression polynomials.
- Time Steps / Year: The number of time steps / year used for the process discretisation.

Product	Model / Engine	Samples	Regression Order	Time Steps / Year
SingleAssetOption(EQ)	BlackScholes / MC	10k	6	na
SingleAssetOption(FX)	BlackScholes / MC	10k	6	na
SingleAssetOption(COMM)	BlackScholes / MC	10k	6	na
MultiAssetOption(EQ)	BlackScholes / MC	10k	2	24
MultiAssetOption(FX)	BlackScholes / MC	10k	2	24
MultiAssetOption(COMM)	BlackScholes / MC	10k	2	24
SingleUnderlyingIrOption	GaussianCam / MC	10k	6	na
MultiUnderlyingIrOption	GaussianCam / MC	10k	2	na
IrHybrid(EQ)	GaussianCam / MC	10k	2	na
IrHybrid(FX)	GaussianCam / MC	10k	2	na
IrHybrid(COMM)	GaussianCam / MC	10k	2	na
IrHybrid(Inf)	GaussianCam / MC	10k	2	na

Table 36: Scripted Trade Pricing Parameters

5.6 Credit Derivatives

5.6.1 Credit Default Swap

CDS are priced by discounting premium payments adjusted for default probability, and the recovery rate. Mid-point approximation is used (i.e. it is assumed the default, if it happens, occurs in the middle of a premium period, in between two premium payments)

The net present value of a CDS from a protection-seller perspective can be expressed:

$$\begin{aligned} NPV =& N \cdot \sum_{i=1}^{n} K \cdot S(t_i) \cdot \delta(t_{i-1}, t_i) \cdot P(t_i) \\ & - N \cdot \sum_{i=1}^{n} \left(LGD - K \frac{\delta(t_{i-1}, t_i)}{2} \right) \cdot (S(t_{i-1}) - S(t_i)) \cdot P\left(t_{i-1} + \frac{t_i - t_{i-1}}{2} \right) \end{aligned}$$

where:

- K: the CDS premium rate
- $S(t_i)$: the survival probability up to cash flow time t_i as seen at valuation time
- $P\left(t_{i-1} + \frac{t_i - t_{i-1}}{2}\right)$: the discount factor at the mid-point between times t_{i-1} and t_i
- $\delta(t_{i-1}, t_i)$: the daycount fraction in years for the period from t_{i-1} to t_i
- LGD: loss given default

The first line of the NPV expression reflects the premium payments weighted with the survival probability for each full period. The second line reflects that if default occurs, the protection seller pays the LGD (times notional), but receives half of the premium (the accrued spread for the time up to default in the middle of the period); the cash flow is discounted back from the middle of the period where the default occurs, and each flow is weighted with the probability of default in this period

$$\text{ProbDefault}(t_{i-1}, t_i) = S(t_{i-1}) - S(t_i).$$

In a standard CDS, the LGD is equal to $1 - R$ where the recovery rate R is determined once a credit event has been deemed to have occured. The recovery rate for a given reference entity and debt tier is determined in a credit event auction.

In a fixed recovery CDS, the recovery rate R is specified at trade inception in the contract terms.

Per ISDA Standard Model, the Real Recovery Rate is used for conversion from ParSpread to Upfront with piecewise constant hazard rate. The Assumed Recovery Rate is used for the conventional spread/upfront conversion using a flat hazard rate within the ISDA standard model. The CDS Real Recovery Rate (also known as the Composite Recovery Rate) is a consensus value that is calculated daily based on observable CDS market data submitted by sell-side bank contributors. It is used for default curve building and pricing except for the conventional spread / upfront conversion mentioned before. The Assumed Recovery Rate (also known as standard recovery rates) is based on a matrix of recovery rate levels defined by ISDA according to Contract Region/ISDA Transaction Type/CDS Tier Recovery factors. These rates do not fluctuate on a daily basis as they reference the ISDA recovery rate matrix fixings.

5.6.2 Asset Backed Credit Default Swap

The pricing of an ABCDS follows the approach of section 5.6.1 closely, with the generalization that changing notional amounts are possible in each period. The ABCDS pricing assumes a deterministic notional schedule that follows the expected notional reductions according to the underlying security's realised and expected future prepayments.

5.6.3 Index Credit Default Swap

There are two pricing approaches for Index Credit Default Swaps. The first approach uses the credit curve of index itself and follow the approach of section 5.6.1. The second approach is a looking through / bottom up approach and calculate the protection leg as the notional weighted sum of the underlying basket constituents using each constituent's credit curve and recovery.

With the second approach we have a natural breakdown of the credit sensitivities to each constituent while on the other hand the pricing is may not consitent to the market (using the index curve). In the index curve approach we can decompose the index sensitivities into the credit sensitivity for each constituent using a weighting schema.

Let $CR01_{Index,T}$ be the credit sensitivities of the index curve on the tenor T, the decomposed sensitivity for constituent $i = 1, ..., n$ is then given by

$$CR01_{i,T} = w_i \, CR01_{Index,T}$$

with $\sum_{j=1}^{n} w_i = 1$.

There are three weighting schemas:

In the *notional weighted* schema the weight is given derived from the notionals N_i of the index constituents, $w_i = \frac{N_i}{\sum_{j=1}^{n} N_j}$.

In the *delta weighted* schema the weight is derived from a simplified credit delta. Assuming a zero bond with maturity of the index cds t_m for each constituent, the approximate credit delta is given by

$$\delta_i = N_i \, t_m \, S_i(t_m)$$

with $S_i(t_m)$ being the survival probability up to maturity of constituent i. The approximated credit delta will be normalized to weights $w_i = \frac{\delta_i}{\sum_{j=1}^{n} \delta_j}$.

The third schema *expected loss weighted* uses as similar approach as the *delta weighted* schema, but instead of the normalized credit delta is uses the expected loss

$$EL_i = N_i \, (1 \, - \, S_i(t_m)) \, (1 \, - \, RR_i).$$

5.6.4 CDS Option

The CDS Option payoff can be written

$$NPV_{CDSO}(t_e) = [NPV_{CDS}(t_e)]^+$$

where $NPV_{CDS}(t_e)$ is the present value of the underlying CDS on exercise date t_e with specified reference credit, CDS maturity and contractual CDS spread (strike).

CDS Options are priced, in analogy to European Swaptions, using the Black model. Following the European Swaption pricing approach, we write the time t price of a forward CDS (starting at time $t_s > t$ and maturing on time $t_m > t_s$)

$$NPV_{CDS}(t) = \omega \, (S(t) - K) \, A(t; t_s, t_m)$$

in terms of running spread K, fair forward CDS spread $S(t)$ and risky annuity $A(t; t_s, t_m)$

$$S(t) = \frac{1}{A(t; t_s, t_m)} \times \text{Default Leg NPV}$$
$$A(t; t_s, t_m) = \frac{1}{K} \times (\text{Coupon Leg NPV} + \text{Accrual NPV})$$

(see section 5.6.1). The option price in the risky annuity measure is then

$$NPV_{CDSO}(t) = A(t; t_s, t_m) \; \mathbb{E}_t^A \left[(\omega(S(T) - K))^+ \right]$$

Assuming log-normal dynamics for thr forward CDS spread $S(t)$ we arrive at a Black formula for today's CDS Option price

$$NPV_{CDSO} = N \sum_{i=1}^{n} \text{Black}(K, S(t_0), \sigma_{t_s,t_m} \sqrt{t_s}, \omega) \cdot A(t_s, t_m)$$

where:

- N: notional
- K: CDS (running) premium
- $S(t_0)$: today's forward CDS spread for CDS start t_s and maturity t_m
- σ_{t_s,t_m}: volatility of the forward CDS spread corresponding to the start t_s and maturity t_m of the period

- ω: 1 for a protection buyer (premium payer), -1 for a protection seller
- $A(t_s, t_m)$: risky annuity for the period from t_s to t_m as of today,

$$A(t_s, t_m) = \sum_{i=s+1}^{m} \delta_i \, P(t_i) \, S(t_i)$$

- δ_i: the daycount fraction in years for the period from t_{i-1} to t_i
- $P(t_i)$: the discount factor for time t_i
- $S(t_i)$: the survival probability for time t_i

See Black Model, Section 6.2, for more details.

If the option underlying contains an upfront spread, this is converted into a corresponding running spread adjustment.

If the CDS Option contract contains *front end protection* and the underlying CDS is a protection buyer, then we add the value contribution

$$N \cdot LGD \cdot (1 - S(t_s)) \cdot P(t_s)$$

Apart from discount curve, default/survival probability curves and LGDs, CDS Option pricing requires Black volatilities which may be available only for index CDS Options for a limited range of indices and short expiries.

5.6.5 Index CDS Option

An Index CDS Option is priced using either

- a Black-Formula approach or
- a numerical integration approach.

The numerical integration approach avoids some approximations that are applied in the Black-Formula and is therefore considered to be more accurate in general.

A. Notation of notionals

In the following we distinguish the following notionals

- N_{uf} is the *unfactored* notional, i.e. the notional excluding all defaults. This is also the notional that is used in the trade xml representation.
- N_{td} is the *trade date* notional, i.e. the notional after accounting for defaults between index inception up to the trade date
- N_{vd} is the *valuation date* notional, i.e. the notional after accounting for defaults between index inception up to the valuation date

We start with the description of the calculation of the Front-End Protection Value (FEP). The calculation of FEP and its components FEP_u (unrealized part) and FEP_r (realized part) is identical for both pricing approaches (Black-Formula, numerical integration).

B. Discount Curves

We distinguish two discount curves:

- The overnight curve associated to the underlying swap currency. We denote discount factors on this curve as $P_{OIS}(0,t)$.
- The overnight curve associated to the collateral currency of the option trade. We denote discount factors on this curve as $P_{OIS,t}(0,t)$

Furthermore we introduce the notation $P_*(0,t)$ as follows:

- $P_*(0,t) := P_{OIS}(0,t)$, i.e. using the swap currency OIS curve, if the option trade is cash settled
- $P_*(0,t) := P_{OIS,t}(0,t)$, i.e. using the trade collateral OIS curve, if the option trade is physically settled

C. Calculation of the Front-End Protection Value FEP

The total front-end protection value as of the valuation date is given by the sum of the realized and unrealised FEPs, discounted from the exercise time t_e on the trade collateral curve:

$$\text{FEP} = P_{OIS,t}(0,t_e)(\text{FEP}_r + \text{FEP}_u) \tag{31}$$

The realized FEP_r is given by the sum over realised losses for underlyings that are defaulted as of the valuation date and have an auction date later or qeual to the FEP Start date. The realised loss of an underlying is given by its (pre-default) notional N_i times one minus its (realised) recovery rate RR_i, i.e.

$$\text{FEP}_r = \sum_i (1 - RR_i)N_i \tag{32}$$

where the sum is taken over the defaulted underlyings. The unrealised FEP_u is given by

$$\text{FEP}_u = \sum_i (1 - RR_i)(1 - S_i(t_e))N_i \tag{33}$$

where $S(t_e)$ is the survival probability until the exercise date and the sum is taken over the non-defaulted underlyings. The survival probability $S_i(t_e)$ and recovery rate $RR_i(t_e)$ is either calculated

- using the individual name's credit curve and recovery rate (FepCurve = Underlying in the pricing engine config) or
- using the index credit curve and recovery rate (FepCurve = Index in the pricing engine config)

The pricing proceeds in two different ways depending on the type of the input volatility, which can be a price-volatility or a spread-volatility:

D. Pricing using a price-volatility input

If the input volatility is given as a price volatility, no numerical integration is required, instead a simple application of the Black-Formula can be used in both pricing engines:

If the strike K is given as a price, we adjust for defaults between trade date and valuation date

$$K_P = 1 - \frac{N_{td}}{N_{vd}}(1 - K) \tag{34}$$

since the contractual strike amount is expressed w.r.t. the trade date notional by market convention, which can be higher than the notional at valuation date.

If the strike is given as a spread, the first step is to convert the spread strike K_s to a price strike K_P as follows:

$$K_P = 1 + \frac{N_{td}}{N_{vd}} \cdot RA(K_s) \cdot (c - K_s) \tag{35}$$

where

- $RA(K_s)$ is the risky strike annuity (see below)
- c is the coupon rate of the CDS

Again, the factor N_{td}/N_{vd} again accounts for the fact that the strike amount is calculated using the risky strike annuity on the notional at trade date.

The risky strike annuity $RA(K_s)$ is calculated as follows:

$$RA(K_s) = \frac{\text{NPV}_{K,cpn} + Acc_{cpn}}{N_{vd} \cdot c \cdot S_K(t_e) \cdot P_{OIS}(0, t_e)} \tag{36}$$

where

- N_{vd} is the underlying index CDS notional[11]
- $\text{NPV}_{K,cpn}$ is the premium leg NPV of a *strike CDS* in which the coupon rate is set to the strike K_S. The NPV is calculated on a flat hazard rate curve that prices the strike CDS to zero. We use the swap currency OIS curve $P_{OIS}(0,t)$ for discounting.
- $Acc_{K,cpn}$ is the accrual rebate of the strike CDS
- $S_K(t_e)$ is the survival probability until option expiry t_e computed on the flat hazard rate curve that prices the strike CDS to zero

The FEP-adjusted forward price is calculated as

$$F_p = \frac{1 - \text{NPV}}{N_{vd} \cdot P_*(0, t_e)} - \frac{\text{FEP}}{N_{vd} \cdot P_{OIS,t}(0, t_e)} \tag{37}$$

where NPV is the value of the underlying swap. We use

[11] In fact the choice of the notional does not affect $RA(K_s)$ since it cancels out, so it can be set to an arbitrary value. For numerical reasons it is set to $1/K_s$ in the actual calculation.

- constituent curves and recovery rates (Curve = Underlying in the pricing engine config) or
- index curve and recovery rate (Curve = Index in the pricing engine config)

as default curves / recovery rates. The Index CDS option price is now calculated as

$$\text{NPV}_{opt} = N_{vd} \cdot P_{OIS,t}(0, t_e) \cdot \text{Black}\left(K_P, F_p, \sigma\sqrt{t_e}, \omega\right) \tag{38}$$

where

- σ is the market volatility for expiry t_e and strike K_P
- ω is 1 for a call, -1 for a put

E. Pricing using a spread-volatility input

If the input volatility is a spread-volatility the pricing proceeds depending on the model.

E.1. Black-Formula

The risky annuity RA is given by

$$RA = \frac{\text{NPV}_{cpn} + Acc_{cpn}}{N_{vd} \cdot c \cdot P_*(0, t_e) \cdot S(t_e)} \tag{39}$$

where

- NPV_{cpn} is the premium leg NPV of the CDS. As described above the value is calculated using the constituent curves and recovery rates or the index curve and recovery rate depending on the pricing engine configuration. The discounting curve is the trade collateral curve for physically settled options resp. the swap currency OIS curve for cash settled options.
- Acc_{cpn} is the accrual rebate of the CDS
- $S(t_e)$ is the survival probability until option expiry t_e

The FEP-adjusted forward spread is approximated as a deterministic value

$$S_p = S + \frac{\text{FEP}}{S(t_e) \cdot RA \cdot N_{vd} \cdot P_{OIS,t}(0, t_e)} \tag{40}$$

where S is the fair spread of the underlying index CDS. Note that the numerical integration engine models S_p as a stochastic quantity on the other hand, see 46. The adjusted strike spread K_s' is calculated as

$$K_s' = c + \frac{N_{td}}{N_{vd}} \frac{RA(K_s)}{S(t_e) \cdot RA} (K_s - c) \tag{41}$$

where

- K_s is the spread strike

- $RA(K_s)$ is the risky strike annuity, see 36
- RA is the risky annuity, see 39

As before, the factor N_{td}/N_{vd} accounts for the fact that the strike adjustment payment upon exercise is calculated using the risky strike annuity on the notional at trade date rather than the valuation date.

Finally, the option price is calculated using the Black formula

$$\text{NPV}_{opt} = P_{OIS,t}(0,t_e) \cdot S(t_e) \cdot RA \cdot N_{vd} \cdot \text{Black}\left(K_s', F_p, \sigma\sqrt{t_e}, \omega\right) \tag{42}$$

with a call (protection buyer) / put (protection seller) indicator $\omega = 1, -1$ and a volatility σ corresponding to the strike K and expiry t_e. The factor $P_{OIS,t}(0,t_e)/P_*(0,t_e)$ accounts for possibly different trade collateral and swap currency OIS curves.

We list two shortcomings of the Black engine compared to the numerical integration engine (see below):

- the Black engine does not convert a price strike to a spread strike. Instead an error is thrown.
- the Black engine does not account for defaults between trade and valuation date when reading the pricing volatility from the market spread volatility surface

E.2. Numerical Integration

The first step is to compute the strike adjustment payment K^*. In case the trade strike is given as a price strike K_P this is

$$K^* = \frac{N_{td}}{N_{vd}}(K_P - 1) \tag{43}$$

and if the trade strike is given as a spread strike K_s this is

$$K^* = \frac{N_{td}}{N_{vd}} \cdot RA(K_s) \cdot (c - K_s) \tag{44}$$

where $RA(K_s)$ is the risky strike annuity, see 36. As before the factor N_{td}/N_{vd} accounts for the market convention that the strike adjustment payment should be based on the trade date notional rather than valuation date notional.

To get the spread strike K_s *adjusted for defaults between trade date and valuation date* we use a numerical root finder to solve

$$RA(K_s) \cdot (c - K_s) = K^* \tag{45}$$

for K_s. Notice that we can skip the numerical root finding if the trade strike is given as a spread strike *and* $N_{td} = N_{vd}$ (in which case K_s is simply the trade strike).

The FEP-adjusted forward price F_p is calculated as in 37. The FEP adjusted forward spread S_p modeled as a lognormal variable

$$S_p(x) = me^{-\frac{1}{2}\sigma^2 t_e + x\sigma\sqrt{t_e}} \tag{46}$$

where σ is the pricing volatility for strike K_s and $x \sim N(0,1)$ and m is a free parameter which will be calibrated in a moment. Compare 46 to the Black pricing approach 40 where we approximated S_p by a deterministic value.

The FEP adjusted index value $V_c(S_p)$ as a function of the FEP-adjusted forward spread S_p is then given as

$$V_c = (S_p - c) \cdot a \tag{47}$$

We approximate the true risky annuity by a continuous risky annuity a to speed up calculations. The continuous risky annuity is defined as

$$a = \int_{t_e}^{t_m} e^{-\int_{t_e}^{u}(r+\lambda)ds} du \tag{48}$$

with interest rate r and instantaneous hazard rate λ. We set r to the average interest rate over the time interval $[t_e, t_m]$ from option expiry to underlying maturity and appxoimate λ by the usual "credit triangle" $S/(1-R)$ where R is the index recovery rate (used to bootstrap the index CDS curve). This yields

$$r_A = \frac{\ln\left(\frac{P_*(0,t_m)}{P_*(t_e)}\right)}{t_e - t_m} \tag{49}$$

$$w = \left(\frac{S}{1-R} + r\right) \cdot (t_m - t_e) \tag{50}$$

$$a = (t_m - t_e)\,\frac{1 - e^{-w}}{w} \tag{51}$$

For small w, $|w| < 10^{-6}$ we use an expansion to compute a to avoid numerical problems due to a being close to $0/0$:

$$a = (t_m - t_e)\left(1 - \frac{1}{2}w + \frac{1}{6}w^2 - \frac{1}{24}w^3\right) \tag{52}$$

The free parameter m from 46 is calibrated to F_p (see 37) by solving

$$\int_{-\infty}^{\infty} V_c(x, m)\,\phi(x)\,dx = 1 - F_p \tag{53}$$

for m numerically. The integration is carried out using the Simpson method with integration bounds $-10 < x < 10$ (i.e. we cover 10 standard deviations of the state variable x in the integration).

The exercise boundary argument x^* is then computed by numerically solving

$$\left(V_c + K^* + \frac{\text{FEP}_r}{N_{vd}}\right)\phi(x) = 0 \tag{54}$$

for x. Finally the option value is computed as

$$\text{NPV}_{opt} = N_{vd} \cdot P_{OIS,t}(0,t_e) \int_{x_L}^{x_U} \omega \cdot \left(V_c + K^* + \frac{\text{FEP}_r}{N_{vd}}\right) \cdot \phi(x)dx \tag{55}$$

with $K_L = x^*, K_U = \infty$ for a call option ($\omega = 1$) or $K_L = -\infty, K_U = x^*$ for a put option ($\omega = -1$). As before, ∞ is replaced by 10 for the actual numerical integration.

F. Pricing Additional Results

Table 37 summarizes the available additional results generated in the various pricing engines and how they map to the variables used in the previous sections.

Variable	Label
FEP_r	realisedFEP
FEP_u	unrealisedFEP
FEP	discountedFEP
N_{td}	tradeDateNotional
N_{vd}	valuationDateNotional
K_P	strikePriceDefaultAdjusted
$RA(K_S)/P_{OIS}(0,t_e)$	forwardRiskyAnnuityStrike
F_P	fepAdjustedForwardPrice
F_P	fepAdjustedForwardSpread
RA	riskyAnnuity
K_s'	adjustedStrikeSpread
K_s	strikeSpread
K^*	strikeAdjustment
$P_{OIS}(0,t_e)$	discountToExerciseTradeCollateral
$P_{OIS}(0,t_e)$	discountToExerciseSwapCurrency
$S(x^*)$	exerciseBoundary
$\int \ldots dx$	integrationGrid
$V_c(x)$	defaultAdjustedIndexValue

Table 37: additional results in index cds option pricing engines.

5.6.6 Synthetic CDO

A synthetic CDS is priced in analogy to a CDS as the difference between a premium leg value and a protection leg value. Key is the calculating of the *expected tranche loss* which affects both the protection leg flow and valuation, and remaining tranche notional that applies to the premium leg valuation.

The total loss given default of the basket up to time t is given by

$$L(t) = \sum_{i=1}^{N} L^i(t), \qquad L^i(t) = (1 - R_i)\, N_i\, 1_{\tau_i < t}$$

Where N is the number of assets, and $1_{\tau_i<t}$ is the indicator function that is equal to one if the time to default τ_i for asset number i is less than t. $L(t)$ is a random variable as the default times τ_i. The tranche loss is then given by

$$L_T(t) = \min(L(t), D) - \min(L(t), A) = \begin{cases} 0, & L(t) \leq A \\ L(t) - A, & A < L(t) < D \\ D - A, & L(t) \geq D \end{cases}$$

The *expected tranche loss* is then calculated by integrating the tranche loss $L_T(t)$ over the probability density function $\rho(x)$ for loss $L(t) = x$ which can be shown to yield

$$\begin{aligned} E[L_T(t)] &= \int_0^\infty dx \left(\min(x, D) - \min(x, A)\right) \rho(x) \\ &= \int_A^D dx \, Pr\{L(t) > x\} \end{aligned}$$

and which shows that only *excess loss probability* $Pr\{L(t) > x\}$ is required to evaluate expected losses. The present value of the expected payoff in time interval $[t_1, t_2]$ is then given by

$$NPV_{t_1,t_2} = \left(E[L_T(t_2)] - E[L_T(t_1)]\right) \, P((t_1 + t_2)/2),$$

and the present value of the entire protection leg by

$$NPV_1 = \sum_{i=1}^{N} \left(E[L_T(t_i)] - E[L_T(t_{i-1})]\right) \, P((t_i + t_{i-1})/2)$$

where $P(t)$ is the discount factor for maturity t. Note that we assume that losses occur equally throughout the period so that discounting at the mid point is justified.

The premium is paid on the protected notional amount, initially $D - A$. This notional is reduced by the expected protection payments E_i at times t_i so that the premium value is calculated as

$$NPV_2 = m \cdot \sum_{i=1}^{N} \left(D - A - E[L_T(t_i)]\right) \cdot \delta_{i-1,i} \cdot P(t_i)$$

where m is the premium rate, $\delta_{i-1,i}$ is the day count fraction between date/time t_{i-1} and t_i. The total CDO price for the protection seller is therefore
$NPV = NPV_2 - NPV_1$.

To compute the expected tranche loss for a basket that shows correlated defaults we use a single-factor Gaussian Copula model.

In the Copula approach, the individual assets' probability distributions are glued together into a multivariate distribution that introduces correlated default events. The Copula approach ensures by construction that the marginal distributions recover each asset's probability of default.

Let $Q_i(t)$ be the cumulative probability that asset i will default before time t, i.e. the time of draw $t_i < t$. Evaluated at the horizon T, $Q_i(T) = p_i$.

In a one-factor Copula model, consider random variables

$$Y_i = a_i\, M + \sqrt{1 - a_i^2}\, Z_i$$

where M and Z_i have independent zero-mean unit-variance distributions and $-1 \leq a_i \leq 1$. The correlation between Y_i and Y_j is then $a_i a_j$.

Let $F_i(y)$ be the cumulative distribution function of Y_i. y is mapped to t such that percentiles match, i.e. $F_i(y) = Q_i(t)$ or $y = F_i^{-1}(Q_i(t))$.

Now let $H_i(z)$ be the cumulated distribution function of Z_i. For given realization of M, this determines the distribution of y:

$$\mathbb{P}(y_i < y|M) = H\left(\frac{y - a_i\, M}{\sqrt{1 - a_i^2}}\right)$$

or

$$\mathbb{P}(t_i < t|M) = H\left(\frac{F_i^{-1}(Q_i(t)) - a_i\, M}{\sqrt{1 - a_i^2}}\right)$$

Let $G(m)$ be the cumulative distribution function of M, so that $g(m) = -\partial_m G(m)$ is its density: Integrating over the density $g(m)$ yields the marginal probability of default for asset i

$$Q_i(t) = \int_{-\infty}^{\infty} \mathbb{P}(t_i < t|M)\, g(m)\, dm = \int H\left(\frac{F_i^{-1}(Q_i(t)) - a_i\, m}{\sqrt{1 - a_i^2}}\right) g(m)\, dm.$$

The loss distribution is now evaluated as follows:

- For each realization m of M: evaluate the probability $p_i = Q_i(T)$ of default for each contract and determine the distribution of independent losses (conditional on M; for this purpose we use in general the *Hull-White Bucketing Algorithm* [44] as e.g. implemented in [49] which allows for different individual p_i's and loss associated loss amounts; this yields $P_k(m)$ for all buckets k
- Average the outcomes $P_k(m)$ for each bucket k over the distribution of M

$$P_k = \int_{-\infty}^{\infty} P_k(m)\, g(m)\, dm.$$

In a Gaussian copula model, G and H are set to the cumulative standard normal distribution. The integral above is evaluated numerically, realizations of M need not be generated with Monte Carlo simulation. Note that the Gaussian distribution is stable under convolution, i.e. also the distribution F of Y is Gaussian.

Apart from discount curve, survival/default probability curves and LGDs for each name in the basket we need copula correlations to price CDOs. These are quoted for equity tranches (attachment point $A = 0$) on indices such as iTraxx or CDX in the market, so-called *base correlations*. Any tranche can be priced as the difference of two equity tranches with different detachment points and respective base correlations.

One-Factor Gaussian Copula Model with Stochastic Recovery

To match IHS Markit methodology and prices we use a stochastic recovery extension of the single-factor Gaussian Copula model above. The model is described in detail in [45]. Here we provide a brief summary, aligning with the notation above:

Choose a discrete (marginal) recovery distribution for each entity i conditioned on default ($\tau_i < t$)

$$\widetilde{R}_i(t) = \begin{cases} r_{i,1} & \text{with probability} \quad p_{i,1} \\ r_{i,2} & \text{with probability} \quad p_{i,2} \\ \vdots & \vdots \quad\quad\quad\quad \vdots \\ r_{i,J} & \text{with probability} \quad p_{i,J} \end{cases}$$

with $r_{i,1} > r_{i,2} > \cdots > r_{i,J}$ such that average/expected recovery rates match the quoted recovery rates:

$$\sum_{j=1}^{J} p_{i,j}\, r_{i,j} = R_i(t) \qquad \text{and} \qquad \sum_{j=1}^{J} p_{i,j} = 1.$$

Now let $Q_{i,j}(t)$ be the cumulative probability of default of entity i until t with recovery rate less than $r_{i,j}$:

$$Q_{i,j}(t) = Q_i(t) \cdot \sum_{k=j+1}^{J} p_{i,k} = Q_i(t) \cdot \left(1 - \sum_{k=1}^{j} p_{i,k}\right)$$

so that

$$0 = Q_{i,J} < Q_{i,J-1} < Q_{i,J-2} < \cdots < Q_{i,0} = Q_i.$$

Generalized thresholds for the factor Y_i driving entity i's default and its recovery rate are now defined as

$$C_{i,j}(t) = N^{-1}(Q_{i,j}(t))$$

with

$$-\infty = C_{i,J} < C_{i,J-1} < C_{i,J-2} < \cdots < C_{i,1} < C_{i,0}.$$

Entity i is considered defaulted if variable Y_i is less than threshold $C_{i,0}$ and it moreover defaults with recovery rate $r_{i,j}$ if $C_{i,j} < Y_i \leq C_{i,j-1}$.

$$r_{i,J} < r_{i,J-1} < r_{i,J-2} < \cdots < r_{i,1}$$

so that the highest recovery rate $r_{i,1}$ (lowest LGD) is associated with the region close to the default threshold $C_{i,0}$, as shown in figure 3.

The Gaussian assumption for Y_i, Z_i and M then leads to the following probabilities conditional on the common factor M:

- Probability of default

$$P_i(M) := \mathbb{P}(t_i \leq t|M) = \mathbb{P}(Y_i \leq C_{i,0}(t)|M) = N\left(\frac{C_{i,0}(t) - a_i\, M}{\sqrt{1 - a_i^2}}\right)$$

  as in the standard Gaussian Copula case

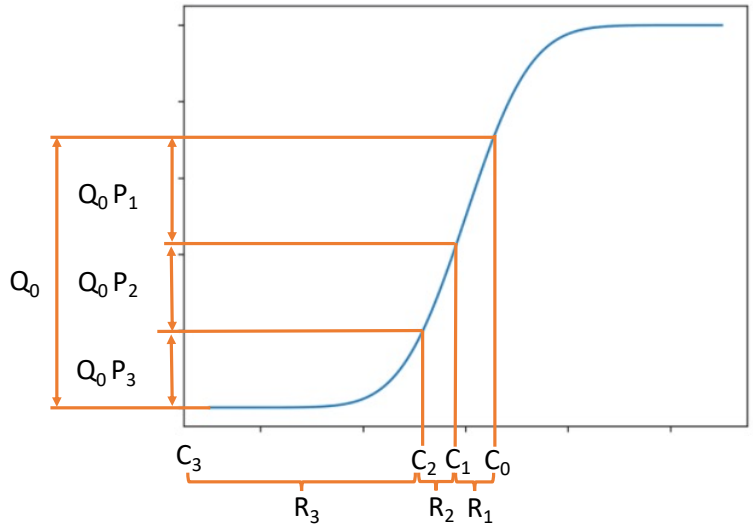

Figure 3: Probabilities, thresholds and recovery rates.

- Probability of recovery rate r_{ij} given default of entity i

$$P_{i,j}(M) := \frac{\mathbb{P}(C_{i,j}(t) < Y_i \leq C_{i,j-1}(t)|M)}{\mathbb{P}(Y_i \leq C_{i,0}(t)|M)} = \frac{N\left(\frac{C_{i,j-1}(t)-a_i\,M}{\sqrt{1-a_i^2}}\right) - N\left(\frac{C_{i,j}(t)-a_i\,M}{\sqrt{1-a_i^2}}\right)}{N\left(\frac{C_{i,0}(t)-a_i\,M}{\sqrt{1-a_i^2}}\right)}$$

- Probability of default with recovery rate r_{ij}

$$P_{i,j}(M) \times P_i(M) = N\left(\frac{C_{i,j-1}(t) - a_i\,M}{\sqrt{1-a_i^2}}\right) - N\left(\frac{C_{i,j}(t) - a_i\,M}{\sqrt{1-a_i^2}}\right)$$

The distribution of portfolio losses, conditional on M

$$L(M) = \sum_{i=1}^{N} N_i\,(1 - R_i(M))\,P_i(M)$$

is then computed by convolution. As in the single-recovery case we want to use a bucketing algorithm when loss amounts and default probabilities are allowed to differ across entities. The *Hull-White Bucketing Algorithm* in [44] was extended slightly to handle the multi-recovery case.

5.6.7 Calibration of default curves for index tranches

If we calculate the npv of an Index CDS based on the underlying constituent CDS spread curves we notice a basis between the intrisinc value and the quoted index spread. The basis and two methods to adjust the underlying curves to close the basis are described in detail in [67]. Our calibration method is based on the idea of the Forward Default Probability Multiplier.

Instead of a calibration of the underlying curves to all quoted index terms in a iterative method and using a termstructure of adjustment factors, we are using a simpler method with a flat adjustment factor λ.

Let $Q_i(t)$ the survival probability of the i-th constituent at time t. Their adjusted survival probability is given by $\tilde{Q_i}(t) = Q_i(t)^\lambda$.

When we are pricing a index CDS tranche, we set the factor λ so that the intrinsic valuation of the underlying index CDS with the adjusted curves matches the quoted price.

5.6.8 RPA

The premium leg of a RPA is priced by discounting premium payments adjusted for default probability, its net present value can be written

$$F = \sum_{i=1}^{n} f_i P_{OIS}(T_i) S(T_i) + A \tag{56}$$

with

- f_i the future payable fee amounts, $i = 1, \ldots, n$
- T_i the payment time of the fee f_i, $i = 1, \ldots, n$
- $P_{OIS}(t)$ the discount factor on the relevant OIS curve (in a collateralised RPA)
- $S(T_i)$ the survival probability of the reference entity between today and T_i
- A is the value of the fee accruals, see below

If the premium payments are given as simple cashflows, $A = 0$. If on the other hand the premium payments are *coupons* and the RPA agreement specifies the payment of accruals in case of a default event, the value of the fee accruals is computed as

$$A = \sum_{i=1}^{n} a_i P_{OIS}(T_i^{\text{mid}}) P_{\text{def}}(A_i^s, A_i^e) \tag{57}$$

using the mid-point rule, where

- T_i^{mid} is the mid point between
  - A_i^s defined as the greater of the accrual start date and the evaluation date and
  - A_i^e the accrual end date
- a_i are the accruals of f_i between A_i^s and the date corresponding to T_i^{mid}
- $P_{\text{def}}(A_i^s, A_i^e)$ is the probably of default of the reference entity between A_i^s and A_i^e

The protection leg of a RPA is priced using different methods, depending on the structure of the underlying. See section 2.2.56 for the detailed criteria that are used to pick the pricing method depending on the trade xml representation:

- Vanilla Swap: Analytic Pricer using a European swaption representation of the underlying EPE profile
- Strucutred Swap: Numeric LGM Pricer

- Callable Swap / Swaption: Numeric LGM Pricer
- Cross Currency Swap: Analytic Pricer using a FX Option representation of the underlying EPE profile
- Treasury Lock: Numeric LGM Pricer

Analytic Pricer using European swaption representation:

The protection leg is priced as a weighted sum over European swaptions

$$P = \sum_{i=1}^{N} p(1-R) P_{\text{def}}(W_i^s, W_i^e) \nu(W_i) \tag{58}$$

where

- W_i is a swaption with expiry t_i representing the option to enter into the cashflows of the underlying swap with payment date greater than t_i. The swaption is found by matching the NPV, Delta and Gamma of the exercise-into underlying swap within a LGM1F model at a suitable expansion point for the LGM state with the NPV, Delta and Gamma of a standard underlying.
- $P_{\text{def}}(W^s i, W_i^e)$ is the probability of default of the reference entity between W_i^s and W_i^e
- R is the assumed recovery rate, or the fixed recovery rate from the RPA termsheet, if given
- p is the participation rate
- $\nu(W_i)$ is the NPV as of the evaluation date of the swaption W_i

The intervals $[W_i^s, W_i^e]$ and swaption expiries $t_i \in [W_i^s, W_i^e]$, $i = 1, \ldots, N$ are chosen such that the intervals cover the interval between

- the greater of the protection start and evaluation date and
- the earlier of the protection end date and the latest underlying swap payment date

One possible concrete choice in the case of a vanilla fix versus float underlying (with a float coupon frequency being a multiple of the fixed coupon frequency) is

- W_i^s the maximum of the evaluation date, protection start date and ith underlying float accrual start date
- W_i^e the minimum of the protection end date and the ith underlying accrual end date
- t_i is the mid point between suitable dates W_i^s and W_i^e (see below)

where only non-empty periods $[W_i^s, W_i^e]$ are kept.

Notice that the protection leg of a RPA is identical to a protection leg of a CDS with nominal V if $\nu(W_i)$ is replaced by a constant amount V. The total NPV of an RPA is given as

$$\text{NPV}_{\text{RPA}} = F - P \tag{59}$$

for a protection seller position resp.

$$\text{NPV}_{\text{RPA}} = P - F \tag{60}$$

for a protection buyer position.

Analytic Pricer using FX Option representation:

The approach is similar to "Analytic Pricer using European swaption representation", but in this case the exercise-into flows of the underlying are matched with FX Spot Trades using the NPV and FX Delta. This matching can be performed in a model independent way, interest rates are assumed to be deterministic here.

Numeric LGM Pricer

The EPE is computed in a LGM1F model. The approach is very similar to the pricing of bermudan swaptions. The calibration approach is also the same for swap underlyings. For "Treasury Lock" underlying a Coterminal ATM calibration is used.

5.6.9 Credit Linked Swap

A credit linked swap is priced using a discounted cashflow approach assuming independent interest and credit rates.

Payments that are made independent of credit events for the reference CDS are priced just as in a vanilla interest rate swap, see 5.1.1.

The NPV of payments that are only made if no credit event has occured until the payment date are weighted in addition with the survival probability for the payment date. If accruals are settled on the event date, it is assumed that defaults occur on the midpoint of the coupon period, i.e. that (on average) half of the accruals are paid and discounted from the midpoint of the coupon period. This is in analogy to premium payment pricing in CDS, see 5.6.1.

Payments that are made in case of a credit event are priced in analogy to protection payments in CDS, see 5.6.1. The discretisation grid is given by the payment dates of the respective legs, refined by additional points to ensure a minimum number of grid points per year as specified in the pricing engine parameter `TimeStepsPerYear`.

The pricing engine provides the following additional results:

- npv_independent: The NPV of the payments that are made independent of credit events.
- npv_credit_linked: The NPV of the payments that are made only if no credit event has occured until the payment date.
- npv_credit_linked_accruals: The NPV of the accruals of the credit linked payments, if accruals are settled.
- npv_default_payments: The NPV of the payments triggered by a credit event, weighted by $(1 - rr)$ where rr is the applicable recovery rate.

- npv_recovery_payments: The NPV of the payments triggered by a credit event, weighted by rr.

5.7 Commodity Derivatives

5.7.1 Commodity Forward

The commodity forwarding curve is used for respective underlying commodity. The net present value of a commodity forward can be priced discounting the cashflows at maturity as per:

$$NPV = \text{Quantity} \cdot \omega \cdot (F(0,T) - K) \cdot P(T)$$

where:

- Quantity: number of units of the underlying commodity
- K: strike price
- $F(0,T)$: the forward commodity price for maturity T at time 0 (valuation date)
- $P(T)$: the discount factor for maturity time T
- ω: 1 for a forward to buy commodity, -1 for a forward to sell a commodity

5.7.2 European Commodity Option

European Commodity Options are priced using the Black-Scholes model with log-normal volatilities. Volatility structures with smile are taken into account where available. The commodity forwarding curve is used for respective commodity.

The Black-Scholes model assumes that the spot commodity price C follows Geometric Brownian Motion:

$$dC/C = \mu(t)dt + \sigma(t)\,dW$$

The model?s drift is calibrated such that the expected future spot commodity price agrees with today?s fair forward commodity price for time T. This means that the commodity forward rate $F(t,T)$ follows drift-free GBM under the T-forward measure:

$$dF(t,T)) = \sigma(t)\,F(t,T)\,dW$$

This leads a Black76 analytical solution for the present value of a European FX option:

$$NPV = N \cdot \text{Black}(K, F(0,T), \sigma\sqrt{T}, \omega) \cdot ?P(T)$$

where:

- N: notional
- K: strike price
- $F(0,T)$: the forward commodity price for maturity T at time 0 (start)
- σ: the volatility of the commodity forward price
- ω: 1 for a call option, -1 for a put option
- $P(T)$: the discount factor for maturity time T

See Black Model, Section 6, for more details.

5.7.3 Commodity Futures Option

Though Commodity Futures Options can be technically American style (as e.g. in the case of base metal futures options at LME), we treat them as European options following [52].

The option payoff on option exercise date t_e with underlying futures contract expiry $T > t_e$ is

$$NPV(t_e) = N \cdot (\omega(F(t_e, T) - K))^+ .$$

We assume that the futures price follows Geometric Brownian Motion:

$$dF(t, T)) = \sigma(t)\, F(t, T)\, dW.$$

This leads to a Black76 analytical solution for the present value of the Futures Option:

$$NPV = N \cdot \text{Black}(K, F(0, T), \sigma\sqrt{t_e}, \omega) \cdot P(T)$$

where:

- N: notional (units of the underyling futures contract)
- K: strike price
- $F(0, T)$: the futures price for contract expiry T at time 0 (start)
- σ: the volatility of the commodity futures price
- ω: 1 for a call option, -1 for a put option
- $P(T)$: the discount factor for the underlying futures expiry T

See Black Model, Section 6, for more details.

5.7.4 Commodity Swap

As in Section 5.7.1, the commodity forwarding curve is used for respective underlying commodity. The net present value of each calculation period in a Commodity Swap can be computed following [52]:

$$NPV_{Period} = \text{Quantity} \cdot \omega \cdot \left(\frac{1}{n} \sum_{i=1}^{n} F(0, T_i) - K \right) \cdot P(T)$$

where:

- Quantity: number of units of the underlying commodity
- K: strike price
- the sum runs over all fixing dates t_i in the calculation period, $i = 1, ..., n$
- $F(0, T_i)$: the forward commodity price for maturity T_i seen at time 0 (valuation date), T_i is the prompt futures expiry date associated with the fixing date t_i if the Swap references Futures prices, or it is the fixing date t_i if the Swap references Commodity spot prices.
- $P(T)$: the discount factor for settlement date T typically 5 business days after the last fixing date in the period

- ω: +1 for a long Commodity Swap, -1 otherwise

As the payoff can also be rolled up into a single payment we generalize the pricing formula, taking the sum over all fixings dates in all calculation periods

$$NPV_{Total} = \text{Quantity} \cdot \omega \cdot \frac{1}{N} \sum_{i=1}^{N} (F(0, T_i) - K) \cdot P(T_i^{Set})$$

where T_i^{Set} denotes the settlement date associated with each fixing, possibly linked to the following period end or swap maturity.

5.7.5 Commodity Basis Swap

The net present value of each calculation period in a Commodity Basis Swap can be computed following [52], and slightly extending Section 5.7.4 :

$$NPV = \text{Quantity} \cdot \omega \cdot \frac{1}{N} \sum_{i=1}^{n} \left(F^{(1)}(0, T_i) - F^{(2)}(0, T_i) - K \right) \cdot P(T_i^{Set})$$

where:

- Quantity: number of units of the underlying commodity
- K: strike price
- the sum runs over all fixing dates t_i across all calculation periods, $i = 1, ..., N$
- $F^{(1,2)}(0, T_i)$: the forward price of commodity 1 resp. 2 for maturity T_i seen at time 0 (valuation date), T_i is the prompt futures expiry date associated with the fixing date t_i if the Swap references Futures prices, or it is the fixing date t_i if the Swap references Commodity spot prices.
- $P(T_i^{Set})$: the discount factor for settlement date following fixing date T_i, possibly a number of business days (typically five) past the next calculation period end, or past the Swap's final calculation period end if all payouts are rolled up into a single payment
- ω: +1 for a long Commodity Swap, -1 otherwise

5.7.6 Commodity Swaption - Future Settlement Prices

This section describes the European swaption pricing when the underlying swap references commodity future settlement prices. In particular, we consider the case where on each pricing date on the commodity floating leg of the underlying swap, the settlement price of the prompt future contract is observed. The calculations can be easily generalised to a non-prompt future contract.

We assume that there are N future contracts and, following Section 2.2 of [52], we assume that each future contract price follows a driftless lognormal process under the risk neutral measure in the domestic currency

$$dF_\alpha(t) = \sigma_\alpha F_\alpha(t) dW_\alpha(t) \tag{61}$$

where $\alpha = 1, \ldots, N$, $0 \leq t \leq s_\alpha$ and s_α in the expiry date of the α-th future contract. Additionally, we assume that the correlation between any two future price processes is

driven by an instantaneous correlation between their two driving Brownian motions. In particular, we assume that

$$d\langle W_\alpha(t), W_\gamma(t)\rangle = \rho_{\alpha,\gamma} \tag{62}$$

for $1 \leq \alpha < \gamma \leq N$. Instead of providing a full correlation matrix, it is convenient in certain cases to parameterise the correlation. In ORE+, we allow for a parameterisation of the form

$$\rho_{\alpha,\gamma} = \exp\left\{-\beta\left(s_\gamma - s_\alpha\right)\right\} \tag{63}$$

where $\beta \geq 0$ and a value of β equal to 0 gives an instantaneous correlation of 1 between all future contract price processes.

We can now write down some straightforward properties of the future price processes that we will use later

$$F_\alpha(t) = F_\alpha(0)\exp\left\{-\frac{1}{2}\sigma_\alpha^2 t + \sigma_\alpha W_\alpha(t)\right\} \tag{64}$$

$$\mathbb{E}\left[\, F_\alpha^2(t)\,\right] = F_\alpha^2(0)\exp\left\{\sigma_\alpha^2 t\right\} \tag{65}$$

$$\mathbb{E}[\, F_\alpha(t)F_\gamma(t)\,] = F_\alpha(0)F_\gamma(0)\exp\left\{\rho_{\alpha,\gamma}\sigma_\alpha\sigma_\gamma t\right\} \tag{66}$$

We now define the commodity swap underlying the commodity swaption that we are valuing. The commodity swap will exchange a sequence of payments that depend on known fixed prices, the commodity fixed leg, against a sequence of payments that reference the settlement prices of the future contracts, the commodity floating leg. The swap has n calculation periods denoted by $I_1, \ldots, I_n$. Associated with the i-th calculation period, I_i, we have the commodity quantity for the calculation period denoted by N_i, the fixed leg price denoted by K_i and the payment date denoted by τ_i. On the commodity floating leg, each calculation period, I_i, contains n_i pricing dates $t_{i,1}, \ldots, t_{i,n_i}$. On each pricing date, the settlement price of the prompt future contract is observed. To associate a pricing date $t_{i,j}$, $j = 1, \ldots, n_i$ with its prompt future contract we define the function $T : [0, +\infty) \to \{0, 1, \ldots, N\}$. This function takes a pricing date and returns the index of the future contract whose settlement price is to be observed. In particluar, this function also defines the roll date convention which can be specified on commodity swap contracts that have a floating leg with averaging. We now define the floating price associated with each calculation period, I_i, as

$$\frac{1}{n_i}\sum_{j=1}^{n_i} F_{T(t_{i,j})}(t_{i,j}) \tag{67}$$

Note that we can have $n_i = 1$ for $i = 1, \ldots, n$. In this case, we have a non-averaging commodity floating leg. In other words, the commodity future contract settlement price is observed on a single date in the calculation period to determine the floating price for the calculation period. If $n_i > 1$, the calculation period I_i is an averaging period. In general, a commodity swap will be either non-averaging or averaging. In the case of averaging swaps, the pricing dates for a calculation period are generally defined as being every business day in the calculation period. The payoff Π_i on payment date τ_i for a commodity swap defined in this way is given by

$$\Pi_i = \omega\left[\frac{1}{n_i}\sum_{j=1}^{n_i} F_{T(t_{i,j})}(t_{i,j}) - K_i\right] N_i \tag{68}$$

where ω is 1 for a payer swap and -1 for a receiver swap.

We now assume that we have a European swaption on the swap defined above with exercise date t_e where $t_e < t_{1,1}$. We assume in what follows that we have deterministic domestic interest rates with $P(t,T)$ denoting the discount factor at time t for maturity T. The value of the swap, $\hat{V}(t_e)$, at time t_e is therefore given by

$$\begin{aligned}\hat{V}(t_e) &= \omega \sum_{i=1}^{n} P(t_e,\tau_i)\frac{N_i}{n_i}\sum_{j=1}^{n_i}\mathbb{E}\big[\, F_{T(t_{i,j})}(t_{i,j}) \;\big|\; \mathcal{F}_{t_e}\,\big] \\ &\quad - \omega \sum_{i=1}^{n} P(t_e,\tau_i) K_i N_i \\ &= \omega \sum_{i=1}^{n} P(t_e,\tau_i)\frac{N_i}{n_i}\sum_{j=1}^{n_i} F_{T(t_{i,j})}(t_e) - \omega \sum_{i=1}^{n} P(t_e,\tau_i) K_i N_i\end{aligned} \tag{69}$$

We note that the first term in (69) is the value of the swap's floating leg at t_e and the second term is the value of the swap's fixed leg at t_e.

Now, the value of the swaption, $V(0)$, at time zero is given by

$$V(0) = P(0,t_e)\,\mathbb{E}\Big[\,\hat{V}(t_e)^+\,\Big] \tag{70}$$

Monte Carlo simulation would be one option for calculating a value for the expectation in (70).

Following Section 2.7.4.2 of [52], we can calculate an approximate value for the expectation in (70) by replacing the floating leg with a lognormally distributed random variable X whose first and second moments match those of the floating leg. Formally, we define

$$A(t_e) = \sum_{i=1}^{n} P(t_e,\tau_i)\frac{N_i}{n_i}\sum_{j=1}^{n_i} F_{T(t_{i,j})}(t_e) \tag{71}$$

and let $X \sim LN(\mu_X, \sigma_X^2 t_e)$ where we impose the conditions

$$\mathbb{E}[\,A(t_e)\,] = \mathbb{E}[\,X\,] = \exp\left\{\mu_X + \frac{1}{2}\sigma_X^2 t_e\right\} \tag{72}$$

$$\mathbb{V}[\,A(t_e)\,] = \mathbb{V}[\,X\,] = \left[e^{\sigma_X^2 t_e} - 1\right]\mathbb{E}[\,X\,]^2 \tag{73}$$

which yields

$$\sigma_X = \sqrt{\frac{1}{t_e}\ln\frac{\mathbb{E}[\,A^2(t_e)\,]}{\mathbb{E}[\,A(t_e)\,]^2}} \tag{74}$$

$$\mu_X = \ln\mathbb{E}[\,A(t_e)\,] - \frac{1}{2}\sigma_X^2 t_e \tag{75}$$

Now, defining the value of the fixed commodity leg at t_e as

$$K^* = \sum_{i=1}^{n} P(t_e,\tau_i) K_i N_i \tag{76}$$

we can write the approximate value, $\tilde{V}(0)$, of the swaption at time zero as

$$\tilde{V}(0) = P(0,t_e)\,\mathbb{E}\big[\,[\omega\,(X-K^*)]^+\,\big] \tag{77}$$

This is the familiar expectation that appears in the Black-76 model as outlined in Section 6.2. In particular, using the Black function defined in Section 6.2, we have

$$\tilde{V}(0) = P(0,t_e)\text{Black}\left(K^*, \mathbb{E}[\,A(t_e)\,], \sigma_X\sqrt{t_e}, \omega\right) \tag{78}$$

All that remains is to write down the explicit value for $\mathbb{E}[\,A(t_e)\,]$ and $\mathbb{E}[\,A^2(t_e)\,]$ appearing in (74) and (75). It is clear that

$$\mathbb{E}[\,A(t_e)\,] = \sum_{i=1}^{n} P(t_e,\tau_i)\frac{N_i}{n_i}\sum_{j=1}^{n_i} F_{T(t_{i,j})}(0) \tag{79}$$

In order to calculate $\mathbb{E}[\,A^2(t_e)\,]$, we write down an explicit expression for $A^2(t_e)$

$$\begin{aligned} A^2(t_e) = &\sum_{i=1}^{n} P^2(t_e,\tau_i)\frac{N_i^2}{n_i^2}\left[\sum_{j=1}^{n_i} F_{T(t_{i,j})}(t_e)\right]^2 \\ &+ 2\sum_{k=2}^{n}\sum_{i=1}^{k-1} P(t_e,\tau_k)P(t_e,\tau_i)\frac{N_kN_i}{n_kn_i}\left[\sum_{j=1}^{n_k} F_{T(t_{k,j})}(t_e)\sum_{l=1}^{n_i} F_{T(t_{i,l})}(t_e)\right] \end{aligned} \tag{80}$$

which in turn gives

$$\begin{aligned} A^2(t_e) = &\sum_{i=1}^{n} P^2(t_e,\tau_i)\frac{N_i^2}{n_i^2}\left[\sum_{j=1}^{n_i} F^2_{T(t_{i,j})}(t_e)\right. \\ &\left. + 2\sum_{k=2}^{n_i}\sum_{l=1}^{k-1} F_{T(t_{i,k})}(t_e)F_{T(t_{i,l})}(t_e)\right] \\ &+ 2\sum_{k=2}^{n}\sum_{i=1}^{k-1} P(t_e,\tau_k)P(t_e,\tau_i)\frac{N_kN_i}{n_kn_i}\left[\sum_{j=1}^{n_k}\sum_{l=1}^{n_i} F_{T(t_{k,j})}(t_e)F_{T(t_{i,l})}(t_e)\right] \end{aligned} \tag{81}$$

Now, taking the expectation and using (65) and (66), we get

$$\begin{aligned} \mathbb{E}\big[\,A^2(t_e)\,\big] = &\sum_{i=1}^{n} P^2(t_e,\tau_i)\frac{N_i^2}{n_i^2}\left[\sum_{j=1}^{n_i} F^2_{T(t_{i,j})}(0)\exp\left\{\sigma^2_{T(t_{i,j})}t_e\right\}\right. \\ &\left. +2\sum_{k=2}^{n_i}\sum_{l=1}^{k-1} F_{T(t_{i,k})}(0)F_{T(t_{i,l})}(0)\exp\left\{\rho_{T(t_{i,k}),T(t_{i,l})}\sigma_{T(t_{i,k})}\sigma_{T(t_{i,l})}t_e\right\}\right] \\ &+ 2\sum_{k=2}^{n}\sum_{i=1}^{k-1} P(t_e,\tau_k)P(t_e,\tau_i)\frac{N_kN_i}{n_kn_i}\left[\sum_{j=1}^{n_k}\sum_{l=1}^{n_i} F_{T(t_{k,j})}(0)\right. \\ &\left. F_{T(t_{i,l})}(0)\exp\left\{\rho_{T(t_{k,j}),T(t_{i,l})}\sigma_{T(t_{k,j})}\sigma_{T(t_{i,l})}t_e\right\}\right] \end{aligned} \tag{82}$$

FX adjustment

If the underlying swap references a commodity quoted in a currency different from the swaption currency we must adjust (79) and (82) accordingly.

We assume that the FX Spot rate Z follows a Geometric Brownian Motion:

$$dZ/Z = \mu_Z(t)dt + \sigma_Z(t)dW_Z \tag{83}$$

For simplicity we assume that $\sigma_Z = 0$ and the adjustment involves converting each futures price with the FX-forward at the price observation time. For completeness we write out the required adjustments.

The floating price associated with each calculation period, I_i, becomes

$$\frac{1}{n_i}\sum_{j=1}^{n_i} F_{T(t_{i,j})}(t_{i,j})Z(t_{i,j}) \tag{84}$$

And we have:

$$\mathbb{E}[\, F_\alpha(t)Z(t)\,] = F_\alpha(0)Z(0)\exp\{(r_d - r_f)t + \rho_{F_\alpha,Z}\sigma_{F_\alpha}\sigma_Z t\} \tag{85}$$

$$\mathbb{E}\big[\, F_\alpha^2(t)Z^2(t)\,\big] = F_\alpha^2(0)Z^2(0)\exp\big\{2(r_d - r_f)t + \sigma_\alpha^2 t + \sigma_Z^2 t + 4\rho_{F_\alpha,Z}\sigma_{F_\alpha}\sigma_Z t\big\} \tag{86}$$

(69) becomes:

$$\hat{V}(t_e) = \omega\sum_{i=1}^{n} P(t_e,\tau_i)\frac{N_i}{n_i}\sum_{j=1}^{n_i} F_{T(t_{i,j})}(t_e)Z(t_{i,j}) - \omega\sum_{i=1}^{n} P(t_e,\tau_i)K_iN_i \tag{87}$$

As before, we can calculate an approximate value for the expectation in ((87)) by replacing the Floating leg with a lognormally distributed random variable X whose first and second moments match those of the floating leg, with:

$$A(t_e) = \sum_{i=1}^{n} P(t_e,\tau_i)\frac{N_i}{n_i}\sum_{j=1}^{n_i} F_{T(t_{i,j})}(t_e)Z(t_{i,j}) \tag{88}$$

Using (85), with $\sigma_Z = 0$, we get:

$$\begin{aligned}\mathbb{E}[\, A(t_e)\,] &= \sum_{i=1}^{n} P(t_e,\tau_i)\frac{N_i}{n_i}\sum_{j=1}^{n_i} F_{T(t_{i,j})}(0)Z(t_e)\exp\{(r_d(t_e,t_{i,j}) - r_f(t_e,t_{i,j}))(t_{i,j} - t_e)\}\\ &= \sum_{i=1}^{n} P(t_e,\tau_i)\frac{N_i}{n_i}\sum_{j=1}^{n_i} F_{T(t_{i,j})}(0)Z(t_e,t_{i,j})\end{aligned} \tag{89}$$

where $Z(t_1, t_2)$ is projected forward FX rate. And:

$$\begin{aligned}\mathbb{E}\big[\, A^2(t_e)\,\big] = \sum_{i=1}^{n} P^2(t_e,\tau_i)\frac{N_i^2}{n_i^2}\Bigg[\sum_{j=1}^{n_i} F^2_{T(t_{i,j})}(0)Z^2(t_e,t_{i,j})\exp\left\{\sigma^2_{T(t_{i,j})}t_e\right\} \\ +2\sum_{k=2}^{n_i}\sum_{l=1}^{k-1} F_{T(t_{i,k})}(0)Z(0,t_{i,k})F_{T(t_{i,l})}(0)Z(t_e,t_{i,l})\exp\left\{\rho_{T(t_{i,k}),T(t_{i,l})}\sigma_{T(t_{i,k})}\sigma_{T(t_{i,l})}t_e\right\}\Bigg] \\ +\,2\sum_{k=2}^{n}\sum_{i=1}^{k-1} P(t_e,\tau_k)P(t_e,\tau_i)\frac{N_kN_i}{n_kn_i}\Bigg[\sum_{j=1}^{n_k}\sum_{l=1}^{n_i} F_{T(t_{k,j})}(0)Z(t_e,t_{k,j}) \\ F_{T(t_{i,l})}(0)Z(t_e,t_{i,l})\exp\left\{\rho_{T(t_{k,j}),T(t_{i,l})}\sigma_{T(t_{k,j})}\sigma_{T(t_{i,l})}t_e\right\}\Bigg]\end{aligned} \tag{90}$$

5.7.7 Commodity Swaption - Spot Prices

This section describes the European swaption pricing when the underlying swap references a commodity spot price. In particular, we consider the case where on each pricing date on the commodity floating leg of the underlying swap, the spot price of a commodity is observed.

Following Section 2.2.5 of [52], we assume that the spot price process under the risk neutral measure in the domestic currency is given by

$$dS(t) = \left[r(t) - r_f(t)\right]S(t)dt + \sigma_S(t)S(t)dW(t) \tag{91}$$

where $r(t)$, $r_f(t)$ and $\sigma_S(t)$ are deterministic with $r(t)$ being the risk free domestic rate, $r_f(t)$ the commodity convenience yield and $\sigma_S(t)$ the instantaneous volatility. Note that we have

$$P(t,T) = \exp\left\{-\int_t^T r(u)du\right\} \tag{92}$$

for $0 \le t \le T < +\infty$ where $P(t,T)$ is as defined in Section 5.7.6 above.

We can now write down some straightforward properties of the spot price process that we will use later. Solving the SDE for $S(t)$, we get

$$\begin{aligned}S(t_2) = S(t_1)\exp\Bigg\{\int_{t_1}^{t_2} r(u) - r_f(u)du \\ +\int_{t_1}^{t_2}\sigma_S(u)dW(u) - \frac{1}{2}\int_{t_1}^{t_2}\sigma_S^2(u)du\Bigg\}\end{aligned} \tag{93}$$

for $0 \le t_1 \le t_2 < +\infty$. The expected value at t_2 given information up to and including t_1 follows immediately

$$\mathbb{E}[\, S(t_2) \mid \mathcal{F}_{t_1}\,] = S(t_1)\exp\left\{\int_{t_1}^{t_2} r(u) - r_f(u)du\right\} \tag{94}$$

It will also be useful to look at $S^2(t)$ which is given by

$$\begin{aligned}S^2(t_2) = S^2(t_1)\exp\Bigg\{2\int_{t_1}^{t_2} r(u) - r_f(u)du + \int_{t_1}^{t_2}\sigma_S^2(u)du \\ +\int_{t_1}^{t_2} 2\sigma_S(u)dW(u) - \frac{1}{2}\int_{t_1}^{t_2}\left(2\sigma_S(u)\right)^2 du\Bigg\}\end{aligned} \tag{95}$$

with associated conditional expected value

$$\mathbb{E}\big[\, S^2(t_2) \;\big|\; \mathcal{F}_{t_1} \,\big] = S^2(t_1) \exp\left\{ 2 \int_{t_1}^{t_2} r(u) - r_f(u) du + \int_{t_1}^{t_2} \sigma_S^2(u) du \right\} \tag{96}$$

Note that the value of a T-maturity long forward contract on the commodity at time t with strike K is given by

$$V_f(t,T) = P(t,T)\, \mathbb{E}[\, S(T) - K \mid \mathcal{F}_t \,] \tag{97}$$

We denote by $F(t,T)$ the value of K that gives the contract a value of zero at time t. This quantity is the T-maturity forward rate. It is clear from this and (94) that

$$F(t,T) = \mathbb{E}[\, S(T) \mid \mathcal{F}_t \,] = S(t) \exp\left\{ \int_t^T r(u) - r_f(u) du \right\} \tag{98}$$

One final quantity that will be useful in subsequent calculations below is the expectation of the product of commodity prices at two separate times. In particular, using the tower property of conditional expectation, (94) and (96)

$$\begin{aligned}
\mathbb{E}[\, S(t_2)S(t_3) \mid \mathcal{F}_{t_1} \,] &= \mathbb{E}[\, \mathbb{E}[\, S(t_2)S(t_3) \mid \mathcal{F}_{t_2} \,] \mid \mathcal{F}_{t_1} \,] \\
&= \mathbb{E}[\, S(t_2)\, \mathbb{E}[\, S(t_3) \mid \mathcal{F}_{t_2} \,] \mid \mathcal{F}_{t_1} \,] \\
&= \mathbb{E}\left[S^2(t_2) \exp\left\{ \int_{t_2}^{t_3} r(u) - r_f(u) du \right\} \;\middle|\; \mathcal{F}_{t_1} \right] \\
&= F(t_1,t_2) F(t_1,t_3) \exp\left\{ \int_{t_1}^{t_2} \sigma_S^2(u) du \right\}
\end{aligned} \tag{99}$$

for $0 \leq t_1 \leq t_2 \leq t_3 < +\infty$.

Now, as in Section 5.7.6 above, we define the commodity swap underlying the commodity swaption that we are valuing. The commodity swap will exchange a sequence of payments that depend on known fixed prices, the commodity fixed leg, against a sequence of payments that reference the commodity spot price, the commodity floating leg. The swap has n calculation periods denoted by $I_1, \ldots, I_n$. Associated with the i-th calculation period, I_i, we have the commodity quantity for the calculation period denoted by N_i, the fixed leg price denoted by K_i and the payment date denoted by τ_i. On the commodity floating leg, each calculation period, I_i, contains n_i pricing dates $t_{i,1}, \ldots, t_{i,n_i}$. On each pricing date, the commodity spot price is observed. We now define the floating price associated with each calculation period, I_i, as

$$\frac{1}{n_i} \sum_{j=1}^{n_i} S(t_{i,j}) \tag{100}$$

Note that we can have $n_i = 1$ for $i = 1, \ldots, n$. In this case, we have a non-averaging commodity floating leg. In other words, the commodity spot price is observed on a single date in the calculation period to determine the floating price for the calculation period. If $n_i > 1$, the calculation period I_i is an averaging period. In general, a commodity swap will be either non-averaging or averaging. In the case of averaging swaps, the pricing dates for a calculation period are generally defined as being every

business day in the calculation period. The payoff Π_i on payment date τ_i for a commodity swap defined in this way is given by

$$\Pi_i = \omega \left[\frac{1}{n_i} \sum_{j=1}^{n_i} S(t_{i,j}) - K_i \right] N_i \tag{101}$$

where ω is 1 for a payer swap and -1 for a receiver swap.

We now assume that we have a European swaption on the swap defined above with exercise date t_e where $t_e < t_{1,1}$. The value of the swap, $\hat{V}(t_e)$, at time t_e is therefore given by

$$\begin{aligned}
\hat{V}(t_e) &= \omega \sum_{i=1}^{n} P(t_e, \tau_i) \frac{N_i}{n_i} \sum_{j=1}^{n_i} \mathbb{E}[\, S(t_{i,j}) \mid \mathcal{F}_{t_e} \,] \\
&\quad - \omega \sum_{i=1}^{n} P(t_e, \tau_i) K_i N_i \\
&= \omega \sum_{i=1}^{n} P(t_e, \tau_i) \frac{N_i}{n_i} \sum_{j=1}^{n_i} S(t_e) \exp\left\{ \int_{t_e}^{t_{i,j}} r(u) - r_f(u) du \right\} \\
&\quad - \omega \sum_{i=1}^{n} P(t_e, \tau_i) K_i N_i \\
&= \omega \sum_{i=1}^{n} P(t_e, \tau_i) \frac{N_i}{n_i} \sum_{j=1}^{n_i} F(t_e, t_{i,j}) - \omega \sum_{i=1}^{n} P(t_e, \tau_i) K_i N_i
\end{aligned} \tag{102}$$

We note that the first term in (102) is the value of the swap's floating leg at t_e and the second term is the value of the swap's fixed leg at t_e.

Now, the value of the swaption, $V(0)$, at time zero is given by

$$V(0) = P(0, t_e)\, \mathbb{E}\Big[\, \hat{V}(t_e)^+ \,\Big] \tag{103}$$

Monte Carlo simulation would be one option for calculating a value for the expectation in (103). For Monte Carlo simulation, it helps to write $\hat{V}(t_e)$ as

$$\hat{V}(t_e) = \omega \sum_{i=1}^{n} \frac{P(0, \tau_i)}{P(0, t_e)} \left[\frac{1}{n_i} \sum_{j=1}^{n_i} S(t_e) \frac{F(0, t_{i,j})}{F(0, t_e)} - K_i \right] N_i \tag{104}$$

to make it clear that we only need to simulate the commodity spot price at t_e which is straightforward.

Following Section 2.7.4.1 of [52], we can calculate an approximate value for the expectation in (103) by replacing the floating leg with a lognormally distributed random variable X whose first and second moments match those of the floating leg. Formally, we define

$$A(t_e) = \sum_{i=1}^{n} P(t_e, \tau_i) \frac{N_i}{n_i} \sum_{j=1}^{n_i} F(t_e, t_{i,j}) \tag{105}$$

and let $X \sim LN(\mu_X, \sigma_X^2 t_e)$ where we impose the conditions

$$\mathbb{E}[\, A(t_e)\,] = \mathbb{E}[\, X\,] = \exp\left\{\mu_X + \frac{1}{2}\sigma_X^2 t_e\right\} \tag{106}$$

$$\mathbb{V}[\, A(t_e)\,] = \mathbb{V}[\, X\,] = \left[e^{\sigma_X^2 t_e} - 1\right] \mathbb{E}[\, X\,]^2 \tag{107}$$

which yields

$$\sigma_X = \sqrt{\frac{1}{t_e} \ln \frac{\mathbb{E}[\, A^2(t_e)\,]}{\mathbb{E}[\, A(t_e)\,]^2}} \tag{108}$$

$$\mu_X = \ln \mathbb{E}[\, A(t_e)\,] - \frac{1}{2}\sigma_X^2 t_e \tag{109}$$

Now, defining the value of the fixed commodity leg at t_e as

$$K^* = \sum_{i=1}^{n} P(t_e, \tau_i) K_i N_i \tag{110}$$

we can write the approximate value, $\tilde{V}(0)$, of the swaption at time zero as

$$\tilde{V}(0) = P(0, t_e)\, \mathbb{E}\big[\, [\omega\,(X - K^*)]^+\,\big] \tag{111}$$

This is the familiar expectation that appears in the Black-76 model as outlined in Section 6.2. In particular, using the Black function defined in Section 6.2, we have

$$\tilde{V}(0) = P(0, t_e) \text{Black}\left(K^*, \mathbb{E}[\, A(t_e)\,], \sigma_X \sqrt{t_e}, \omega\right) \tag{112}$$

All that remains is to write down the explicit value for $\mathbb{E}[\, A(t_e)\,]$ and $\mathbb{E}[\, A^2(t_e)\,]$ appearing in (108) and (109). For the expectation, we have

$$\begin{aligned} \mathbb{E}[\, A(t_e)\,] &= \sum_{i=1}^{n} P(t_e, \tau_i) \frac{N_i}{n_i} \sum_{j=1}^{n_i} \mathbb{E}[\, S(t_e)\,] \exp\left\{\int_{t_e}^{t_{i,j}} r(u) - r_f(u) du\right\} \\ &= \sum_{i=1}^{n} P(t_e, \tau_i) \frac{N_i}{n_i} \sum_{j=1}^{n_i} F(0, t_{i,j}) \end{aligned} \tag{113}$$

In order to calculate $\mathbb{E}[\, A^2(t_e)\,]$, we write down an explicit expression for $A^2(t_e)$

$$\begin{aligned} A^2(t_e) = &\sum_{i=1}^{n} P^2(t_e, \tau_i) \frac{N_i^2}{n_i^2} \left[\sum_{j=1}^{n_i} F(t_e, t_{i,j})\right]^2 \\ &+ 2 \sum_{k=2}^{n} \sum_{i=1}^{k-1} P(t_e, \tau_k) P(t_e, \tau_i) \frac{N_k N_i}{n_k n_i} \left[\sum_{j=1}^{n_k} F(t_e, t_{k,j}) \sum_{l=1}^{n_i} F(t_e, t_{i,l})\right] \end{aligned} \tag{114}$$

which in turn gives

$$\begin{aligned} A^2(t_e) = &\sum_{i=1}^{n} P^2(t_e, \tau_i) \frac{N_i^2}{n_i^2} \left[\sum_{j=1}^{n_i} F^2(t_e, t_{i,j}) \right. \\ &\left. + 2 \sum_{k=2}^{n_i} \sum_{l=1}^{k-1} F(t_e, t_{i,k}) F(t_e, t_{i,l})\right] \\ &+ 2 \sum_{k=2}^{n} \sum_{i=1}^{k-1} P(t_e, \tau_k) P(t_e, \tau_i) \frac{N_k N_i}{n_k n_i} \left[\sum_{j=1}^{n_k} \sum_{l=1}^{n_i} F(t_e, t_{k,j}) F(t_e, t_{i,l})\right] \end{aligned} \tag{115}$$

Now, taking the expectation and using (96) and (94), we get

$$\begin{aligned}\mathbb{E}\big[\,A^2(t_e)\,\big] &= \sum_{i=1}^{n} P^2(t_e,\tau_i)\frac{N_i^2}{n_i^2}\left[\sum_{j=1}^{n_i} F^2(0,t_{i,j})\exp\left\{\int_0^{t_e}\sigma_S^2(u)du\right\}\right.\\ &\left. +2\sum_{k=2}^{n_i}\sum_{l=1}^{k-1} F(0,t_{i,k})F(0,t_{i,l})\exp\left\{\int_0^{t_e}\sigma_S^2(u)du\right\}\right]\\ &+2\sum_{k=2}^{n}\sum_{i=1}^{k-1} P(t_e,\tau_k)P(t_e,\tau_i)\frac{N_kN_i}{n_kn_i}\left[\sum_{j=1}^{n_k}\sum_{l=1}^{n_i} F(0,t_{k,j})\right.\\ &\left. F(0,t_{i,l})\exp\left\{\int_0^{t_e}\sigma_S^2(u)du\right\}\right]\end{aligned} \tag{116}$$

FX adjustment

Similarly to 5.7.6, if the underlying swap references a commodity quoted in a currency different from the swaption currency, we need to make adjustmentds for the FX rate. This gives us:

$$A(t_e) = \sum_{i=1}^{n} P(t_e,\tau_i)\frac{N_i}{n_i}\sum_{j=1}^{n_i} F(t_e,t_{i,j})Z(t_{i,j}) \tag{117}$$

$$\begin{aligned}\mathbb{E}[\,A(t_e)\,] &= \sum_{i=1}^{n} P(t_e,\tau_i)\frac{N_i}{n_i}\sum_{j=1}^{n_i}\mathbb{E}[\,S(t_e)\,]\exp\left\{\int_{t_e}^{t_{i,j}} r(u)-r_f(u)du\right\}Z(t_e,t_{i,j})\\ &= \sum_{i=1}^{n} P(t_e,\tau_i)\frac{N_i}{n_i}\sum_{j=1}^{n_i} F(0,t_{i,j})Z(t_e,t_{i,j})\end{aligned} \tag{118}$$

$$\begin{aligned}\mathbb{E}\big[\,A^2(t_e)\,\big] &= \sum_{i=1}^{n} P^2(t_e,\tau_i)\frac{N_i^2}{n_i^2}\left[\sum_{j=1}^{n_i} F^2(0,t_{i,j})\exp\left\{\int_0^{t_e}\sigma_S^2(u)du\right\}Z^2(t_e,t_(i,j))\right.\\ &\left. +2\sum_{k=2}^{n_i}\sum_{l=1}^{k-1} F(0,t_{i,k})F(0,t_{i,l})Z(t_e,t_{i,k})Z(t_e,t_{i,l})\exp\left\{\int_0^{t_e}\sigma_S^2(u)du\right\}\right]\\ &+2\sum_{k=2}^{n}\sum_{i=1}^{k-1} P(t_e,\tau_k)P(t_e,\tau_i)\frac{N_kN_i}{n_kn_i}\left[\sum_{j=1}^{n_k}\sum_{l=1}^{n_i} F(0,t_{k,j})\right.\\ &\left. F(0,t_{i,l})\exp\left\{\int_0^{t_e}\sigma_S^2(u)du\right\}Z(t_e,t_{k,j})Z(t_e,t_{i,l})\right]\end{aligned} \tag{119}$$

5.7.8 Commodity Average Price Option - Future Settlement Prices

This section describes the pricing of a Commodity Average Price Option (APO) that references future settlement prices. In particular, we consider the case where on each pricing date in the calculation period of the APO, the settlement price of the prompt future contract is observed. The calculations can be easily generalised to a non-prompt future contract.

We assume that there are N future contracts and that their price processes are as outlined in Section 5.7.6. The APO has a single calculation period containing n pricing dates $t_0 < t_1 < \ldots < t_n$. On each pricing date, the settlement price of the prompt future contract is observed. As in Section 5.7.6, to associate a pricing date t_i, $i = 1, \ldots, n$ with its prompt future contract we define the function $T : [0, +\infty) \to \{0, 1, \ldots, N\}$. This function takes a pricing date and returns the index of the future contract whose settlement price is to be observed. We now define the average price associated with the calculation period as

$$A = \frac{1}{n} \sum_{i=1}^{n} F_{T(t_i)}(t_i) \tag{120}$$

The payoff on payment date $\tau \geq t_n$ for a long commodity APO with quantity N and strike K is then given by

$$[\omega (A - K)]^+ N \tag{121}$$

where ω is 1 for a call and -1 for a put. Now, the value of the APO, $V(0)$, at time zero is given by

$$V(0) = P(0, \tau) \, \mathbb{E}\big[\, [\omega (A - K)]^+ N \, \big] \tag{122}$$

Monte Carlo simulation is one option for calculating the value for the expectation in (122). In particular, we can take the future price option surface if one exists, read off the volatility of the relevant future price processes and simulate the future price processes as defined in (61).

Section 2.7.4.2 of [52], presents a closed form analytical approximation for the expectation in (122). The quantity A in (120) is approximated with a lognormal random variable by matching the first and second moments. Let $X \sim LN(\mu_X, \sigma_X^2 t_n)$ where we impose the conditions

$$\mathbb{E}[\, A \,] = \mathbb{E}[\, X \,] = \exp\left\{ \mu_X + \frac{1}{2} \sigma_X^2 t_n \right\} \tag{123}$$

$$\mathbb{V}[\, A \,] = \mathbb{V}[\, X \,] = \left[e^{\sigma_X^2 t_n} - 1 \right] \mathbb{E}[\, X \,]^2 \tag{124}$$

which yields

$$\sigma_X = \sqrt{\frac{1}{t_n} \ln \frac{\mathbb{E}[\, A^2 \,]}{\mathbb{E}[\, A \,]^2}} \tag{125}$$

$$\mu_X = \ln \mathbb{E}[\, A \,] - \frac{1}{2} \sigma_X^2 t_n \tag{126}$$

With this definition of X, we can now write the approximate value, $\tilde{V}(0)$, of the APO at time zero as

$$\tilde{V}(0) = P(0, \tau) N \, \mathbb{E}\big[\, [\omega (X - K)]^+ \, \big] \tag{127}$$

This is the familiar expectation that appears in the Black-76 model as outlined in Section 6.2. In particular, using the Black function defined in Section 6.2, we have

$$\tilde{V}(0) = P(0, \tau) N \, \text{Black}\left(K, \mathbb{E}[\, A \,], \sigma_X \sqrt{t_n}, \omega\right) \tag{128}$$

All that remains is to write down the explicit value for $\mathbb{E}[\,A\,]$ and $\mathbb{E}[\,A^2\,]$ appearing in (125) and (126). It is clear that

$$\mathbb{E}[\,A\,] = \frac{1}{n}\sum_{i=1}^{n} F_{T(t_i)}(0) \tag{129}$$

For $\mathbb{E}[\,A^2\,]$, we have

$$\begin{aligned}
\mathbb{E}\big[\,A^2\,\big] &= \mathbb{E}\left[\left(\frac{1}{n}\sum_{i=1}^{n} F_{T(t_i)}(t_i)\right)\left(\frac{1}{n}\sum_{j=1}^{n} F_{T(t_j)}(t_j)\right)\right] \\
&= \frac{1}{n^2}\sum_{i,j=1}^{n} \mathbb{E}\big[\,F_{T(t_i)}(t_i)F_{T(t_j)}(t_j)\,\big] \\
&= \frac{1}{n^2}\sum_{i,j=1}^{n} F_{T(t_i)}(0)F_{T(t_j)}(0)\exp\left\{\rho_{T(t_i),T(t_j)}\sigma_{T(t_i)}\sigma_{T(t_j)}t_{\min(i,j)}\right\}
\end{aligned} \tag{130}$$

5.7.9 Commodity Average Price Option - Spot Prices

This section describes the pricing of a Commodity Average Price Option (APO) that references commodity spot prices. In particular, we consider the case where on each pricing date in the calculation period of the APO, the commodity spot price is observed.

We assume that the commodity spot price process is as outlined in Section 5.7.7 and in particular (91). The APO has a single calculation period containing n pricing dates $t_0 < t_1 < \ldots < t_n$. On each pricing date, the commodity spot price is observed. We now define the average price associated with the calculation period as

$$A = \frac{1}{n}\sum_{i=1}^{n} S(t_i) \tag{131}$$

The payoff on payment date $\tau \geq t_n$ for a long commodity APO with quantity N and strike K is then given by

$$[\omega\,(A-K)]^{+}\,N \tag{132}$$

where ω is 1 for a call and -1 for a put. Now, the value of the APO, $V(0)$, at time zero is given by

$$V(0) = P(0,\tau)\,\mathbb{E}\big[\,[\omega\,(A-K)]^{+}\,N\,\big] \tag{133}$$

Monte Carlo simulation is one option for calculating the value for the expectation in (133). In particular, one needs to simulate the spot price process $S(t)$.

Section 2.7.4.1 of [52], presents a closed form analytical approximation for the expectation in (133). The quantity A in (131) is approximated with a lognormal random variable by matching the first and second moments. Let $X \sim LN(\mu_X, \sigma_X^2 t_n)$ where we impose the conditions

$$\mathbb{E}[\,A\,] = \mathbb{E}[\,X\,] = \exp\left\{\mu_X + \frac{1}{2}\sigma_X^2 t_n\right\} \tag{134}$$

$$\mathbb{V}[\,A\,] = \mathbb{V}[\,X\,] = \left[e^{\sigma_X^2 t_n} - 1\right]\mathbb{E}[\,X\,]^2 \tag{135}$$

which yields

$$\sigma_X = \sqrt{\frac{1}{t_n} \ln \frac{\mathbb{E}[\,A^2\,]}{\mathbb{E}[\,A\,]^2}} \tag{136}$$

$$\mu_X = \ln \mathbb{E}[\,A\,] - \frac{1}{2}\sigma_X^2 t_n \tag{137}$$

With this definition of X, we can now write the approximate value, $\tilde{V}(0)$, of the APO at time zero as

$$\tilde{V}(0) = P(0,\tau) N \, \mathbb{E}\big[\,[\omega\,(X-K)]^+\,\big] \tag{138}$$

This is the familiar expectation that appears in the Black-76 model as outlined in Section 6.2. In particular, using the Black function defined in Section 6.2, we have

$$\tilde{V}(0) = P(0,\tau) N \, \text{Black}\left(K, \mathbb{E}[\,A\,], \sigma_X \sqrt{t_n}, \omega\right) \tag{139}$$

All that remains is to write down the explicit value for $\mathbb{E}[\,A\,]$ and $\mathbb{E}[\,A^2\,]$ appearing in (136) and (137). It is clear that

$$\mathbb{E}[\,A\,] = \frac{1}{n}\sum_{i=1}^{n} F(0,t_i) \tag{140}$$

where $F(t,T)$ is as defined in (98). For $\mathbb{E}[\,A^2\,]$, using (99) we have

$$\begin{aligned}
\mathbb{E}\big[\,A^2\,\big] &= \mathbb{E}\left[\left(\frac{1}{n}\sum_{i=1}^{n} S(t_i)\right)\left(\frac{1}{n}\sum_{j=1}^{n} S(t_j)\right)\right] \\
&= \frac{1}{n^2}\sum_{i,j=1}^{n} \mathbb{E}[\,S(t_i)S(t_j)\,] \\
&= \frac{1}{n^2}\sum_{i,j=1}^{n} F(0,t_i)F(0,t_j)\exp\left\{\int_0^{t_{\min(i,j)}} \sigma_S^2(u)du\right\}
\end{aligned} \tag{141}$$

FX adjustment

Similarly to 5.7.6, if the underlying references a commodity quoted in a currency different from the option currency, we need to make adjustmentds for the FX rate. This gives us:

$$\mathbb{E}[\,A\,] = \frac{1}{n}\sum_{i=1}^{n} F(0,t_i)Z(0,t_i) \tag{142}$$

where $F(t,T)$ is as defined in (98). For $\mathbb{E}[\,A^2\,]$, using (99) we have

$$\begin{aligned}
\mathbb{E}\big[\,A^2\,\big] &= \mathbb{E}\left[\left(\frac{1}{n}\sum_{i=1}^{n} S(t_i)Z(t_i)\right)\left(\frac{1}{n}\sum_{j=1}^{n} S(t_j)Z(t_j)\right)\right] \\
&= \frac{1}{n^2}\sum_{i,j=1}^{n} \mathbb{E}[\,S(t_i)Z(t_i)S(t_j)Z(t_j)\,] \\
&= \frac{1}{n^2}\sum_{i,j=1}^{n} F(0,t_i)Z(0,t_i)F(0,t_j)Z(0,t_j)\exp\left\{\int_0^{t_{\min(i,j)}} \sigma_S^2(u)du\right\}
\end{aligned} \tag{143}$$

5.7.10 Commodity Spread Option

European Commodity Spread Options are priced using the Kirk approximation described in Section 2.9.2 of [52].

We assume that the spot commodity prices S^1 and S^2 follows two correlated Geometric Brownian Motion:

$$dS_1/S_1 = \mu_1(t)dt + \sigma_1(t)\, dW_1$$
$$dS_2/S_2 = \mu_2(t)dt + \sigma_2(t)\, dW_2$$

with $\left\langle dW_1, dW_2 \right\rangle = \rho$.

The payoff on payment date for a long commodity spread option with quantity N and strike K is then given by

$$\left[\omega \left(S_1(T) - S_2(T) - K\right)\right]^+ N \tag{144}$$

If $K \ll S_2(T)$ then $S_2(T) + K$ is approximately lognormal distributed and the ratio of two lognormal processes is again lognormal.

Clark defines two new processes:

$$Y(t) = S_2(t) + K \exp(-r(T-t))$$

and

$$Z(t) = \frac{S_2(t)}{Y(t)}.$$

The payoff of the spread option becomes

$$Y(T) \left[\omega \left(Z(T) - 1\right)\right]^+ N \tag{145}$$

If we calibrate the drift so that today's price matches the expected future commodity price we can price this European option with the normal Black76 formula using $Z(0) = \frac{S_1(0)}{S_2(0)+K}$ and $\sigma_Z = \sqrt{\sigma_1^2 + (\sigma_2 \frac{S_2(t)}{Y(t)})^2 + \sigma_1 \sigma_2 \frac{S_2(t)}{Y(t)} \rho}$.

See Black Model, Section 6, for more details.

5.7.11 Commodity Calendar Spread Option - Spot Prices

European Commodity Calendar Spread Options are priced using the Kirk approximation described in Section 2.9.3 of [52].

The payoff on payment date for a long commodity spread option with quantity N and strike K is then given by

$$\left[\omega \left(S(T) - S(t_1) - K\right)\right]^+ N \tag{146}$$

We defining a new process S_2 that has volatility σ from $0 \le t \le t_1$ and zero drift and volatility after t_1.

The total variance of $S_2(T)$ is $\sigma^2 t_1$ which is the same as $\bar{\sigma}^2 T$ with $\bar{\sigma} = \sigma * \sqrt{\frac{t1}{T}}$.

Under the assumption that this product is not path dependent and we can express the payout as:

$$[\omega (S(T) - S_2(T) - K)]^+ N \tag{147}$$

We can use the Kirk approximation with $S_1 = S$, $\sigma_1 = \sigma$ and S_2 defined as above with $\sigma_2 = \bar{\sigma} = \sigma * \sqrt{\frac{t1}{T}}$.

5.7.12 Commodity Calendar Spread Option - Future Prices

If we model Futures the observed contracts will be the prompt future at time t_1 and T. We can use the Kirk approximation but with the initial asset prices of the those futures and their corresponding volatilities.

5.7.13 Commodity Asian Spread Option

Let $A^f_{t_n,t_{n_1}}$ the arithmetic average price of the underlying f between t_n and t_{n_1} .

The payoff of a Asian spread option of difference between two average prices is:

$$\left[\omega \left(A^f_{t_n,t_{n_1}} - A^g_{t_m,t_{m_1}} - K\right)\right]^+ N \tag{148}$$

After applying the adjustements describes in Section 2.7.4 of [52] for average price options (see 5.7.8) one can use the transformed volatilities in Kirk approximation as described for non-averaging options. If the payoff is a calendar spread of two averages of different future contract months but the same underlying asset the near end volatility needs to be scaled by $\sqrt{\frac{t_1}{T}}$ as described in 5.7.11.

5.7.14 Commodity Variance and Volatility Swap

Commodity Variance Swaps and volatility swaps are priced using a replicating portfolio method [66]. See section 5.4.12 for the equivalent pricing of an Equity Variance Swap.

The pricing engine set up is, in analogy to Equity Variance Swaps:

5.8 Bond Derivatives

5.8.1 Forward Bond

A forward bond is a derivative contract entered at time t with the agreement to purchase at time T_f an underlying bond contract whose maturity is T. Clearly at inception $t \leq T_f \leq T$. $P(t_1, t_2)$, $t_1 \leq t_2$ is a discount factor from t_2 to t_1. For the purpose of this description notional is set to 1.

1. Default-free forward bond in multi-curve framework:
   (a) If $t < T_f$ forward contract is derivative. Hence the OIS curve is used for discounting of the derivative product.

Listing 242: "Robust" Commodity Variance Swap pricing engine configuration.

```
<Product type="CommodityVarianceSwap">
  <Model>BlackScholesMerton</Model>
  <ModelParameters/>
  <Engine>ReplicatingVarianceSwapEngine</Engine>
  <EngineParameters>
    <Parameter name="Scheme">Segment</Parameter>
    <Parameter name="Bounds">PriceThreshold</Parameter>
    <Parameter name="Steps">1000</Parameter>
    <Parameter name="PriceThreshold">1E-10</Parameter>
    <Parameter name="MaxPriceThresholdSteps">500</Parameter>
    <Parameter name="PriceThresholdStep">0.1</Parameter>
  </EngineParameters>
</Product>
```

(b) If $t > T_f$ ordinary bond is hold but the buyer of the bond. Hence the BondReferenceYield (BRY) curve is used for discounting. A *LiquiditySpread* / *ValuationSpread* can be applied on top of the BRY curve.

(c) For compounding a compounding curve (COM) is used and discounting is done via the Bond Reference Yield Curve (BRY)

Forward bond valuation:

- Today's price of *restricted* bond, with cashflows only counted for $t_i > T_f$:

$$V(t) = \sum_{T_f < t_i < T} c_{t_i} P_{BRY}(t, t_i) + P_{BRY}(t, T).$$

  The sum is over cashflows that occur after the date of entry into a forward contract. It reflects the coupons of the restricted bond. The term $P_{BRY}(t, T)$ reflects the redemption payment, c_{t_i} reflects a coupon payment at time t_i.

- Cash value at maturity of forward (in return to bond):

$$Cash(T_f) = V(t) * P_{COM}(t, T_f)^{-1}$$

  Notice that this value is compounded according to a COM curve with respective discount factor $P_{COM}(t, T_f)$.

- Let K be the agreed price of the bond (to be paid at T_f). This price can be given either as a *dirty* or *clean* price. The difference between dirty and clean price is taken account of by appropriate subtraction of an accrual factor in case of the clean price.
  Value of forward contract at t:

$$F(t, T_f) = (Cash(T_f) - K) * P_{OIS}(t, T)$$

  $P_{OIS}(t, T)$ is used here for discounting as risk of the derivative (forward bond) is assumed to be covered.

- In praxis both P_{COM} and P_{BRY} will often be chosen as RePo curve.

2. Forward on default-able bond:

   The basic pricing is as described under point *1)* but now we account for the fact that at T_f a bond is delivered that *might default*. Default might occur before or after maturity T_f of the forward. In the first case a defaulted bond would be delivered at T_f.
   Forward on default-able bond valuation proposal:

   (a) Conditioned on the assumption that no default event occurs before forward maturity:

   - Today's price of *restricted* default-able bond, with cashflows only counted for $t_i > T_f$:

$$V(t) = \sum_{T_f < t_i < T} c_{t_i} \bar{P}_{BRY}(t, t_i) + \bar{P}_{BRY}(t, T) + R(T_f, T)$$

   – R reflects face-value recovery between T_f and T:

$$R(T_f, T) = R \int_{T_f}^{T} \rho(s) P_{BRY}(T_f, s) * Spread \; ds,$$

   where ρ is the time-density of default conditioned on the event that no default occurs before T_f.

   – $\bar{P}_{BRY} = P_{BRY} * Spread * CondSurvivalProb$ accounts for potential default

   – Conditional survival probabilities are given by

$$CondSurvivalProb(t, t_i) = S(t, t_i)/S(t, T_f)$$

   – $S(t_1, t_2)$ being the survival probability between t_1 and t_2.

   – BRY: base reference curve, COM: compounding curve

   - As before we set

$$Cash(T_f) := \frac{V(t)}{P_{COM}(t, T_f)}$$

   (b) Taking account of possible default before T_f. If default occurs before T_f we receive a defaulted bond. In other words we only receive recovery value, but we will have to still pay K. The NPV of the derivative under face value recovery is therefore

$$\begin{aligned} F(t, T_f) = &\int_t^{T_f} (RN - K) * P_{OIS}(t, s) \rho(s) ds \\ &+ (Cash(T_f) - K) * S(t, T_f) * P_{OIS}(t, T_f) \end{aligned}$$

   Notice that now the OIS curve is used for discounting as we value a derivative.

The cashflow reports provides the relevant future cashflows of the underlying bond with the corresponding discount factors from the BRY and the forward cashflows at expiry date with the discount factor from the OIS curve. The sum of the present values of the foward cashflows is equal to the Forward NPV.

The additional results provided together with the cashflows and NPV are described in table 38.

Result Label	Description
bondCashflow	The amounts of the future cashflows of the underlying bond, not discounted as a vector.
bondCashflowPayDates	The corresponding payment dates of the underlying bond cashflows.
bondCashflowSurvivalProbabilities	The survival probabilities corresponding to the underlying bond cashflows.
bondCashflowDiscountFactors	The discount factors from the BRY corresponding to the underlying bond cashflows.
bondRecovery	The expected recovery from the underlying bond in case of a default.
forwardBondCashflow	The amounts of the contract Value $F(t, T_f) = (Cash(T_f) - K)$ and premium payments
forwardBondCashflowPayDates	The corresponding payment dates of the forward cashflows.
forwardBondCashflowSurvivalProbabilities	The survival probabilities corresponding to the forward cashflows.
forwardBondCashflowDiscountFactors	The discount factors from the OIS corresponding to the forward cashflows.
forwardBondRecovery	The expected payoff of the forward in case of a default before T_f.

Table 38: Additional Results Bond Forward

5.8.2 Bond Total Return Swap

A total return swap is a derivative contract entered at time t in which one counterparty pays out the total returns of an underlying asset and receives a regular fixed or floating cash flow from the other counterparty. In this section total return swaps with underlying bond are described. The total return of the bond is comprised of

- coupon, redemption and amortization payments of the bond, including recovery payments in case of default
- compensation payments that reflect changes of the clean bond value along the TRS schedule.

Let T denote the maturity of the underlying bond. $P(t_1, t_2)$, $t_1 \leq t_2$ is a discount factor from t_2 to t_1. To highlight that P belongs to a certain curve a subscript with the curve's name will be added, e.g.P_{BRY} is the discount factor corresponding to the Bond Reference Yield curve. For the purpose of this description notional is assumed to be 1. Furthermore for the sake of simplicity of this description we assume that the swap is not composite, i.e. the return is paid in the bond currency.

Hence the value of the ordinary bond (with payment schedule $\{t_i\}_i$, coupons c_{t_i} and assuming notional-recovery) in Bond currency is given by

$$V(t) = \sum_{t_i > t} c_{t_i} \bar{P}(t, t_i) + \bar{P}(t, T) + R(t, T) \tag{149}$$

where

- $\bar{P}(t, t_i) = P_Y(t, t_i) \cdot S(t, t_i)$ is the combined discount factor that accounts for reference yield, bond specific liquidity and potential default, with $P_Y(t, t_i) = P_{BRY}(t, t_i) \cdot P_L(t, t_i)$
- $P_{BRY}(t, t_i)$ is the discount factor of the chosen bond reference yield curve
- $P_L(t, t_i)$ is the discount factor reflecting a bond-specific liquidity spread adjustment z_i to the reference yield curve above, i.e. $P_L(t, t_i) = \exp(-z_i(t - t_i))$ where z_i may also be constant/flat
- $S(t_1, t_2)$ is the survival probability between t_1 and t_2.
- $R(t, T)$ reflects face-value recovery between t and T:

$$R(t, T) = R \int_t^T \rho(s)\, P_Y(t, s)\, ds$$

  where ρ is the time-density of default and $P_Y(t, t_i) = P_{BRY}(t, t_i) \cdot P_L(t, t_i)$.

Note that the liquidity spread adjustment factor $P_L(t, t_i)$ above is used to match quoted prices of liquid bonds, given the bond reference yield curve $P_{BRY}(t, t_i)$ and the bond survival probability curve $S(t, t_i)$.

In the pricing of the TRS we will also encounter the valuation of a *restricted bond*, i.e. a bond whose cashflows are restricted to occur after a certain time T_f. Today's price of the restricted default-able bond is

$$V_{T_f}(t) = \sum_{T_f < t_i < T} c_{t_i} \bar{P}(t, t_i) + \bar{P}(t, T) + R(T_f, T).$$

where $R(T_f, T)$ reflects face-value recovery between T_f and T:

$$R(T_f, T) = R \int_{T_f}^T \rho(s)\, P_Y(T_f, s)\, ds$$

To compute the change of bond value between t and T_f we must forecast the value of the bond at the future time T_f, this is similar to the valuation of forward bonds, section 5.8.1.

In the multicurve framework the following discount curves will be relevant:

- P_Y: curve for discounting the bond cashflows, typically a Repo curve
- P_D: the curve used for discounting derivative cash flows taking the CSA currency into account, see section 7

The discounted compensation payment for the time interval $[t_n, t_{n+1}]$ is given by

$$\begin{aligned} C(t_n, t_{n+1}) = &\left(\frac{V_{t_{n+1}}(t)}{P_Y(t, t_{n+1})} - AccruedAmount(t_{n+1}) \right) \times P_D(t, t_{n+1}) \\ &- \left(\frac{V_{t_n}(t)}{P_Y(t, t_n)} - AccruedAmount(t_n) \right) \times P_D(t, t_{n+1}) \end{aligned}$$

This means we compute clean Bond prices as of both observation dates at period start and end (this is achieved by compouding the time-t prices to the respective period dates using the Bond yield curve); the cash flow at period end is then given by the clean price difference, and the cash flow is finally discounted to time t using the derivative discount curve.

In case dirty instead of clean prices are usd to compute the compensation payments the above formula is amended in the obvious way, in this case we do not subtract the accrued amount as of t_{n+1} resp. t_n.

The NPV of the long derivative contract is then given by

$$NPV(t) = \sum_n C(t_n, t_{n+1}) + V'(t) - NPV_{FundingLeg}(t).$$

where

- $\{t_n\}_n$ is the schedule of the compensation payments;
- $V'(t)$ is the value of the bond cash flows as in (149) but replacing $P_Y(t, t_i) \to P_D(t, t_i)$ in the dicount factors $\bar{P}(t, t_i)$ since we are valuing the bond cash flows now in a derivative context, i.e. $\bar{P}(t, t_i) \to P_D(t, t_i)\, S(t, t_i)$ still taking potential default into account; if the bond maturity is later than the swap maturity, cashflows after the swap maturity are excluded from the valuation and the recovery value is only computed until the swap maturity in this context.
- $NPV_{FundingLeg}(t)$ is the value of the TRS pay leg which is assumed here to be a vanilla fixed or floating leg.

5.8.3 Bond Option

Bond Options are European options and usually valued using the Black's model, which assumes that the value of the underlying bond at exercise time T in the future has a lognormal distribution. The pricing formula for a Bond Option that *knocks out* if the bond defaults before the option expiry is

$$\begin{aligned} NPV^{call}_{\text{knock-out}} &= (1-p)P(0,T)\left[F_B N(d_1) - K N(d_2)\right] \\ NPV^{put}_{\text{knock-out}} &= (1-p)P(0,T)\left[K N(-d_2) - F_B N(-d_1)\right] \\ d_1 &= \frac{\ln(F_B/K) + \sigma_B^2 T/2}{\sigma_B \sqrt{T}} \\ d_2 &= d_1 - \sigma_B \sqrt{T} \end{aligned}$$

where F_B is the forward bond price, σ_B is the price volatility, K is the strike price of the bond option, T is its time to option maturity, $P(0,T)$ is the (risk-free) discount

factor for maturity T, and p is the default probability of the bond until the option expiry. If a yield volatility σ_y is given, the price volatility is derived as

$$\sigma_B = \sigma_Y D y \tag{150}$$

if σ_y is a lognormal volatility and

$$\sigma_B = \sigma_Y D(y + s) \tag{151}$$

if σ_y is a shifted lognormal volatility with shift s and

$$\sigma_B = \sigma_Y D \tag{152}$$

if σ_Y is a normal volatility. Here D denotes the forward modified duration and y the forward yield of the bond w.r.t. F_B.

The forward bond price F_B is computed as for a plain bond but a) only taking cashflows into account that are paid after the option expiry and b) using forward discount factors on the underlying's reference curve (including a security spread, if given) and forward survival probabilities computed from the underlying's credit curve as of the option expiry date, i.e. the forward price is computed *conditional on survival* of the underlying until the option expiry.

If the option *does not knock out* if the bonds defaults before the option expiry, the forward bond price used to calculate the option price in the Black formula above is replaced by the weighted sum of the forward bond price conditional on survival until the option expiry weighted with the survival probability of the bond until the option eypiry and the recovery value of the bond weighted with the default probability of the bond until option expiry, i.e. in this case we have

$$\begin{aligned}
F_{B,\text{no knock-out}} &= (1-p)F_B + pR \\
NPV^{call}_{\text{no knock-out}} &= P(0,T)\left[F_{B,\text{no knock-out}}N(d_1) - KN(d_2)\right] \\
NPV^{put}_{\text{no knock-out}} &= P(0,T)\left[KN(-d_2) - F_{B,\text{no knock-out}}N(-d_1)\right] \\
d_1 &= \frac{\ln(F_{B,\text{no knock-out}}/K) + \sigma_B^2 T/2}{\sigma_B\sqrt{T}} \\
d_2 &= d_1 - \sigma_B\sqrt{T}
\end{aligned}$$

where R is the recovery value of the bond in case of a default before option expiry. The forward modified duration D and forward yield y is computed w.r.t. $F_{B,\text{no knock-out}}$ in this case.

The cashflow reports provide the relevant future cashflows of the underlying bond and the non-discounted option npv at expiry date. The sum of the column present value of the underlying bond cashflows is equal to the price of the forward Bond F_B.

The additional results provided together with the cashflows and NPV are described in table 39.

Result Label	Description
knockOutProbability	The probability p, that the bond defaults before option expiry .
CashStrike	The strike in terms of cash.
FwdCashPrice	The forward cash price of the bond, $(1-p)F_B + pR$.
PriceVol	The volatility used.
ExpectedBondRecovery	The expected recovery from the underlying bond in case of defaulting before expiry (pR).

Table 39: Additional Results Bond Option

5.9 Hybrid Trades

5.9.1 Composite Trade

Composite Trades are priced as the sum of their component trades. As such, the pricing model of the composite trade is a combination of the pricing models of each trade:

$$NPV = \sum_i NPV_i$$

where:

- NPV_i: is the NPV of each component trade, i.

5.9.2 Generic Total Return Swap / Contract for Difference (CFD)

A generic total return swap is priced using the accrual method, i.e.

- The return leg NPV is computed as the difference between the price of the underlying asset today and the price at the valuation start date of the current return period. If the asset currency is not equal to the swap currency, both prices are converted to the swap currency before taking the difference, using the FX rates at the valuation start date and today, respectively. If the underlying asset has paid cashflows (dividends, coupons, amortisation payments) in the current return period, the sum of these cashflows is added to the return leg NPV (converted to the swap currency using today's FX rate).
- The funding leg NPV is computed as the accrued interest of the current coupon period.
- If any additional future cashflows are given, their NPV is the sum of their amounts, converted to the swap currency using today's FX rate.

The swap NPV is the difference between the return and the funding leg (if the return is received) or vice versa (if the return is paid). The additional results provided together with the NPV are described in table 40. In terms of these result labels, the NPV of the swap is the sum of assetLegNpv, fundingLegNpv and additionalCashflowLegNpv.

Result Label	Description
underlyingCurrency	The currency of the underlying.
returnCurrency	The currency of the return payments.
fundingCurrency	The currency of the funding payments.
pnlCurrency	The currency in which the pnl is expressed, this is always the funding currency.
s0	The value of the underlying at the start of the current valuation period, in underlyingCurrency.
s1	The value of the underlying today, in underlyingCurrency.
fx0	The FX conversion factor from underlyingCurrency to returnCurrency at the start of the current valuation period. If initialPrice is given and in returnCurrency, the value will be 1 instead
fx1	The FX conversion factor from underlyingCurrency to returnCurrency at the end of the current valuation period.
assetLegNpv	The npv of the return leg in return currency. This is s1 x fx1 - s0 x fx0 + underlyingCashflows x fx1 if the leg is received.
fundingLegNpv	The npv of the funding leg in funding currency . This is the accrued amount on the funding leg (for a unit notional) x fundingLegNotionalFactor x fundingLegFxFactor, if the leg is received.
fundingLegFxFactor	The FX conversion factor to the funding currency at the start of the funding leg period.
fundingLegNotionalFactor	If an initialPrice is given for this funding period, initialPrice x underlyingMultiplier. Otherwise, the underlying price at the start of this funding leg period multiplied by UnderlyingMultiplier.
underlyingCashflows	Sum of underlying cashflows paid in the current valuation period (up to and including today), in underlyingCurrency. These can be bond cashflows (for bond underlyings) or dividends (for equity basket underlyings).
underlyingMultiplier	For (convertible) bond underlyings, the number of bonds (BondNotional), for equity (option) baskets the quantity.
additionalCashflowLegNpv	The sum of future additional cashflows in additional cashflows currency.
currentNotional	The current period's notional in pnl currency. This is s0 x fx0 x fx(returnCcy → pnlCcy).
fxConversionAssetLegNpvToPnlCurrency	Today's FX conversion factor from return currency to pnl currency.
fxConversionAdditionalCashflowLegNpvToPnlCurrency	Today's FX conversion factor from additional cashflows currency to pnl currency

Table 40: Additional Results Generic Total Return Swap Pricing

Rationale for the accrual method

If the swap contains the right to unwind the position on a daily basis, the accrual method is an obvious choice. In the following we give a rationale for this method even ignoring these unwind rights to highlight the differences to a full discounting valuation approach:

Consider one return period with pricing dates $t_1 < t_2$. The return leg pays (assuming it is the receiver leg)

$$S(t_2) - S(t_1) \tag{153}$$

on t_2. Here, $S(t)$ denotes the underlying asset price, including all income (e.g. dividends) and - in case of a composite swap - converted to the return currency using the prevailing market FX spot rate $X(t)$. The funding leg pays (the minus sign indicating that it is the payer leg)

$$-\left(S(t_1)e^{\int_{t_1}^{t_2} r(s)ds} - S(t_1)\right) \tag{154}$$

on t_2 where $r(s)$ is the interest short rate. Notice that the funded amount is $S(t_1)$ due to the resetting nature of a total return swap. Furthermore, we assume $r(s)$ is determinisitc and the funding rate is consistent with the market rate, i.e. in practical terms we fund "at Libor flat" and there is no fixing risk.

Let's add cashflows to the return leg

$$-S(t_1) \text{ at } t_1, S(t_1) \text{ at } t_2 \tag{155}$$

and to the funding leg

$$S(t_1) \text{ at } t_1, -S(t_1) \text{ at } t_2 \tag{156}$$

which cancel out, i.e. they do not change the trade. The return leg now has cashflows

$$-S(t_1) \text{ at } t_1, S(t_2) \text{ at } t_2 \tag{157}$$

and the funding leg has cashflows

$$S(t_1) \text{ at } t_1, -S(t_1)e^{\int_{t_1}^{t_2} r(s)ds} \text{ at } t_2 \tag{158}$$

Both legs are obviously worth zero at t_1. The valuation of a total return swap therefore reduces to the valuation of the cashflows for the current return period with pricing dates $t_1 < t < t_2$.

On the return leg we have a future payment $S(t_2)$ at t_2 which evaluates to $S(t)$. On the funding leg we have a future payment $-S(t_1)e^{\int_{t_1}^{t_2} r(s)ds}$ which evaluates to $-S(t_1)e^{\int_{t_1}^{t} r(s)ds}$. cashflows $-S(t_1)$ to the return leg and $S(t_1)$ to the funding leg, both paid at t we see that the return leg value can be written as

$$S(t) - S(t_1) \tag{159}$$

and the funding leg value as

$$-\left(S(t_1)e^{\int_{t_1}^{t} r(s)ds} - S(t_1)\right) \tag{160}$$

The latter expression represents the accrued amount on the funding leg between t_1 and t, which is apprxoimated by the linear accrued amount of the current funding coupon in the practical computation. This shows that under the assumptions stated above the valuation of a total return swap under the accrual method is equivalent to the usual no-arbitrage valuation.

5.10 Cash Products

5.10.1 Bond

A vanilla bond is priced by discounting its future coupon and notional payments. The discounting is done on a reference curve (typically a Libor curve in the bond currency) plus possibly a security specific spread. In addition default risk can be taken into account by weighting the payments with their respective survival probability and adding a recovery value reflecting the value of a recovery payment in case of the default of the issuer.

If there are liquid price quotes available for a bond, the security spread can be implied to match this price given a reference curve and (possibly) a default curve.

5.10.2 Bond Repo

The cash leg of a bond repo is priced using either a mark to market or accrual approach. In the mark to market approach the outstanding future interest and notional payments are discounted on a rate curve build from quoted repo instruments (assuming these are available). In the accrual approach the price of the cash leg is defined as the outstanding notional plus accrued interests.

The security leg of a bond repo is priced as a cash bond, i.e. the present value of this leg is the dirty price of the bond provided as collateral.

The total price of the bond repo can either be defined as the difference of the security leg and the cash leg (repo) or vice versa (reverse repo) or also as the price of the cash leg alone. In the former case the price reflects the value of the un- or over-collateralized part of the transaction, in the latter case the price reflects the value of the future cash leg payments alone.

5.10.3 Convertible Bond

A convertible bond is priced in a jump diffusion model, see [25].

Basics

The model dynamics for the stock price $S(t)$ is given by

$$dS/S(t^-) = (r(t) - q(t) + \eta h(t, S(t^-)))dt + \sigma(t)dW(t) - \eta dN(t) \tag{161}$$

with a risk free rate $r(t)$, a continuous dividend yield $q(t)$, a default intensity $h(t, S)$, a volatility $\sigma(t)$, a default loss fraction for the equity $\eta \in [0, 1]$ and a Cox process $N(t)$ with

$$E_t(dN(t)) = h(t, S(t^-))dt \tag{162}$$

The notation $S(t^-)$ is shorthand for $\lim_{\epsilon \downarrow 0} S(t - \epsilon)$. The first jump of $N(t)$ represents the default of the equity. See equation (1) in [25]. We support a local default intensity of the form

$$h(t, S(t)) = h_0(t) \left(\frac{S(0)}{S(t)} \right)^p \tag{163}$$

with a deterministic function $h_0(t)$ that is independent from $S(t)$ and a parameter $p \geq 0$. The parameters p and η can be set in the pricing engine configuration.

Both $h_0(t)$ and $\sigma(t)$ are calibrated to market data, i.e. to a market default curve and a market equity volatility surface. In general this requires the numerical evolution of the density of S_t via its Fokker-Planck PDE. The corner case $p = 0$ allows for a simplified treatment and faster calibration of the model.

The value of a derivative $V = V(t, S)$ is computed by solving the PDE corresponding to the SDE 161. We do this in terms of the log equity price, i.e. we introduce a new variable $z = \ln(S)$, and write $v(t, z) = V(t, e^z)$, see also (6) in [25]:

$$v_t + \left(r(t) - q(t) + \eta h(t, e^z) - \frac{1}{2}\sigma(t)^2 \right) v_z + \frac{1}{2}\sigma(t)^2 v_{zz} \tag{164}$$

$$-(r(t) + h(t, e^z))v + h(t, e^z)R(t, e^z) = 0 \tag{165}$$

Here, $R(t, e^z)$ is a recovery term paid in case of the equity default, which may depend on time t and the *pre-default* stock price e^z.

Non-Exchangeables Recovery Term

For non-exchangeables, the default of the equity is the same as the default of the bond issuer. We assume a deterministic, constant recovery rate ρ for the bond in terms of the (oustanding) nominal N_t . We assume that in case of the issuer default the investor has a claim on the higher of ρN_t and the conversion value on the recovered equity price $C_t^R S(t)(1 - \eta)$, i.e. we have

$$R(t, e^z) = \max\{\rho N_t, C_t^R e^z (1 - \eta)\} \tag{166}$$

For $\eta = 1$ this simplifies to $R = R(t) = \rho N_t$, independent of the equity price $S = e^z$. Conversion is always possible in the case of a default, even if no voluntary conversion

right could be exercised e.g. due to a contingent conversion clause at the time of default.

Exchangeables Recovery Term

For exchangeables, we have to consider two sources of default risk, the bond issuer default risk and the equity issuer default risk. We denote the associated default intensities by $h^B(t, e^z)$ for the bond issuer and $h^S(t, e^z)$ for the equity issuer. The pricing PDE 164 is modified to

$$v_t + \left(r(t) - q(t) + \eta h^S(t, e^z) - \frac{1}{2}\sigma(t)^2\right) v_z + \frac{1}{2}\sigma(t)^2 v_{zz} \tag{167}$$

$$-(r(t) + h^S(t, e^z) + h^B(t, e^z))v \tag{168}$$

$$+h^S(t, e^z)R^S(t, e^z) + h^B(t, e^z)R^B(t, e^z) = 0 \tag{169}$$

For non-secured exchangeables,

- in case of the bond issuer default, the investor has a claim on the bond recovery value only, i.e.
$$R^B(t, e^z) = \rho N_t$$
- in case of the equity issuer default, the convertible bond transaction terminates, but the investor has still a claim on the full notional of the bond, i.e.
$$R^S(t, e^z) = N_t$$

For secured exchangeables,

- in case of the bond issuer default, the investor has a claim on the (non-defaulted) equity conversion value plus the recovery value on the bond notional less the conversion value, i.e.
$$R^B(t, e^z) = C_t^R e^z + \max(\rho(N_t - C_t^R e^z), 0)$$
- in case of the equity issuer default, the convertible bond transaction terminates, but the investor has still a claim on the full notional of the bond, i.e.
$$R^S(t, e^z) = N_t$$

Calibration of h_0 and σ to market data

We fix a set of calibration times $t_1, \ldots, t_n$ and assume h_0 and σ to be piecewise flat w.r.t. this time grid. Our aim is to bootstrap the functions h_0 and σ so that market values for defaultable bonds and ATM equity options are matched by model prices for these instruments, i.e. for the defaultable bond we have the condition

$$P_D^{mkt}(0, t_i) = P_D^{mdl}(0, t_i) = E\left(e^{-\int_0^t r(s)ds} 1_{\tau > t_i}\right) \tag{170}$$

for $i = 1, \ldots, n$, where $P_D^{mkt}(0, t_i)$ is the market value of a defaultable bond paying 1 at t_i and $P^{mdl}(0, t_i)$ is the model value of this bond with τ denoting the model default

time (first jump time). The market value $P_D^{mkt}(0,t_i)$ is computed as the product of the risk free zero bond price and the survival probability up to time t_i. Both the zero bond price and the survival probability are computed on the rate and credit curves stripped from market instruments in the usual way. The condition for the equity option reads

$$C^{mkt}(K,t_i) = C^{mdl}(K,t_i) = E\left(e^{-\int_0^t r(s)ds} 1_{\tau>t_i}(S-K)^+\right) \tag{171}$$

where $C^{mkt}(K,t_i)$ is the price of an equity call option at strike K and with maturity t_i, computed on the market equity forward curve and volatility surface. $C^{mdl}(K,t_i)$ is the corresponding model price.

In general we follow the calibration procedure outlined in [25], section 4.2 with a 2-dimensional minimization of the sum of squares of relative pricing errors for defaultable zero bonds and ATM equity options at each calibration time t_i. The Fokker-Planck equation for the transition density $p(t,z,s,y)$ for times $s \geq t$ is

$$p_s - \frac{1}{2}p_{yy}\sigma(s,e^y)^2 + \tag{172}$$

$$\left(r(s) - q(s) + \eta h(s,e^y) - \frac{1}{2}\sigma(s,e^y)^2\right)p_y + \tag{173}$$

$$\left(r + h(s,e^y) + h_y(s,e^y)\right)p = 0 \tag{174}$$

also compare [25], equation 11, where we assume σ independent of y in our setup (at least for the time being). For our parametrisation

$$h(s,e^y) = h_0(s)(S(0)/e^y)^p \tag{175}$$

we have

$$h_y(s,e^y) = -ph(s,e^y) \tag{176}$$

For $p=0$ we can use a simplified approach for the bootstrap: In this case, the piecewise hazard rate h_0 can be directly set to

$$h_0(t) = \frac{-\ln(S(t_i)/S(t_{i-1}))}{t_i - t_{i-1}} \tag{177}$$

for $t_{i-1} \leq t < t_i$. The model volatility σ can be set to

$$\sigma(t) = \sqrt{V^2 t_i - \sum_{j=0}^{i-1}\sigma(t_j)^2(t_{j+1}-t_j)} \tag{178}$$

for $t_{i-1} \leq t < t_i$. Here, V is computed as the implied Black volatility of the market option premium $C(K,t_i)$ weighted with the market survival probability $S(0,t_i)$ and using an adjusted forward level F^*

$$F^* = \frac{F}{S(0,t_i)^\eta} \tag{179}$$

where F is the market equity forward level.

Notice that for exchangeables, the equity-linked market default curve is used to calibrate $h(s,e^y) = h^S(s,e^y)$ in 172. The bond-linked term $h^B(t,e^z)$ in the pricing PDE 167 on the other hand is directly derived from the bond-linked market default curve and has no impact on the model calibration.

For cross-currency the model calibration is done in the target equity currency, i.e. *after* switching from S^* to S, see the next section on this.

Cross Currency

If the stock denominates in a different currency ccy1 than the bond ccy2, we proceed as follows. Let S^* be the original equity price process in ccy1, i.e. (compare 161):

$$dS^*/S^*(t^-) = (r_{ccy1}(t) - q(t) + \eta h^*(t, S^*(t^-)))dt + \sigma(t)dW(t) - \eta dN(t) \tag{180}$$

where we index $r(t)$ by the equity currency for clarity. We assume the FX rate process X converting one unit of ccy1 into ccy2 to follow a lognormal dynamics

$$dX/X = (r_{ccy2}(t) - r_{ccy1}(t))dt + \sigma_X(t)dW_X(t) \tag{181}$$

with risk free rates r_{ccy2} and r_{ccy1} in ccy2 (domestic currency) resp. ccy1 (foreign currency). The instantenous correlation between FX and equity process is assumed to be ρ, i.e.

$$dW dW_X = \rho dt \tag{182}$$

We now set $S := XS^*$, which is a synthetic equity denominated in ccy2 and work out its dynamics. By Ito,

$$dS = d(XS^*) = XdS^* + S^*dX + dXdS^* \tag{183}$$

so after some simplifications

$$dS/S = (r_{ccy2} - q + \rho\sigma\sigma_X + \eta h(t, S(t^-)))dt + \sqrt{\sigma^2 + \sigma_X^2 + 2\rho\sigma\sigma_X}d\tilde{W} - \eta dN(t) \tag{184}$$

Note that we replaced $h^*(t,S^*)$ with a new function $h(t,S)$. We will parametrise this new $h(t,S)$ as $h(t,S) = h_0(t)(S_0/S)^p$ to obtain the model for S. This dynamics is still in the ccy1 measure. Changing the measure to ccy2 gives the final dynamics

$$dS/S = (r_{ccy2} - q + \eta h(t, S(t^-)))dt + \sqrt{\sigma^2 + \sigma_X^2 + 2\rho\sigma\sigma_X}d\tilde{W} - \eta dN(t) \tag{185}$$

i.e. we can proceed an in the single currency case with a modified equity volatility $\sqrt{\sigma^2 + \sigma_X^2 + 2\rho\sigma\sigma_X}$.

Perpetuals

Perpetuals are priced by cutting off the maturity at the evaluation date plus a period specified in the pricing engine parameters of the Bond trade type under "OpenEndDateReplacement". A typical setting for this value is 30Y or 50Y. Notice that the parameter is read from the Bond trade type configuration rather than the ConvertibleBond trade type configuration, even for convertible bonds.

A perpetual convertible bond does not pay any final redemption at its virtual maturity date.

Dividend Handling

Currently, we support a continuous dividend model. The absolute dividend amount D paid between $0 < t_1 < t_2$ as seen from t_2 is estimated as

$$D = S_{t_2}\left(e^{\int_{t_1}^{t_2} q(s)ds} - 1\right) \tag{186}$$

Historical dividends with ex-dividend date before the evaluation date are added to this value, as appropriate.

N-of-M triggers

N-of-M triggers are priced following the implied barrier aprroach proposed in Jasper Anderluh and Hans van der Weide: Parisian Options – The Implied Barrier Concept, M. Bubak et al. (Eds.): ICCS 2004, LNCS 3039, pp. 851–858, 2004. Springer-Verlag Berlin Heidelberg 2004.

Curves used in practice (benchmark curve, equity forecast)

In practice we use a benchmark curve b to discount the bond value instead of the equity forecast curve r. We modify the pricing PDE 164, 167 in a straightforward way by replacing r with b in the discounting term, i.e. the former PDE becomes

$$v_t + \left(r(t) - q(t) + \eta h(t, e^z) - \frac{1}{2}\sigma(t)^2\right) v_z + \frac{1}{2}\sigma(t)^2 v_{zz} \tag{187}$$

$$-(b(t) + h(t, e^z))v + h(t, e^z)R(t, e^z) = 0 \tag{188}$$

and likewise for the latter PDE. The benchmark curve also includes a security spread, if defined. Notice that we do not modify the Fokker-Planck PDE 172 for model calibration, i.e. in the calibration step we use r as the discounting curve always.

Moreover, for the cross-currency case we have a drift term for the original equity S^*

$$r - q + \eta h \tag{189}$$

Here, r (equity forecast curve) is not necessarily identical to r_{ccy1} (xccy-consistent discounting curve for ccy1). Therefore the drift for the currency-changed S is given by

$$r + (r_{ccy2} - r_{ccy1}) - q + \eta h \tag{190}$$

If $r = r_{ccy1}$ this expression obviously collapsed to 185.

Model Flavour Selection

There are some pricing engine parameters that control the model behaviour on a fundamental level, see table 41. These flags can be used to mimick the behaviour of other vendor models.

Numerical Implementation of the Calibration

The model is calibrated on a configurable set of times

$$t_1, \ldots, t_m \tag{191}$$

specified under `Bootstrap.CalibrationGrid` in the pricing engine configuration. All tenors from the specified grid before the maturity date of the convertible bond are kept and the maturity date itself is added to the resulting grid to avoid calibration for times beyond the bond maturity and at the same time ensuring that we do not need to extrapolate model functions beyond the last calibration point in the pricing.

The case $p = 0$ does not require any particular numerical methods, the piecewise flat model functions h_0 and σ can be directly computed as described in formulae 177 and 178.

For $p > 0$ the solution of the Fokker-Planck PDE is numerically solved using a discretisation of the state variable $y(t) = \ln(S(t))$

$$y_1, \ldots, y_n \tag{192}$$

The grid geometry is determined by the following parameters (configurable in the pricing engine configuration with parameter labels given in brackets):

- the number n of grid points (`Bootstrap.StateGridPoints`)
- an optional mesher concentration m_c to generate a non-uniform grid (`Bootstrap.MesherConcentration`)
- a mesher epsilon m_ϵ (`Bootstrap.MesherEpsilon`)
- a mesher scaling m_s (`Bootstrap.MesherScaling`)

The minimum value y_1 and maximum value y_n covered by the grid are determined as follows. First, the time original calibration time grid 191 is enriched by additional points to ensure that a certain given number of time steps per year (specified as `Boostrap.TimeStepsPerYear`) is used for the numerical solution of the PDE later on, i.e.

$$t_1, \ldots, t_m \to t'_1, \ldots, t'_M \tag{193}$$

with the refined time grid t'_i and $M \geq m$. On each grid point t'_i we compute the forward equity price $F(t'_i)$ for $i = 1, \ldots, M$ and set

$$y_1 = \ln\left(\min_{i=1}^{M} F(t_i)\right) - m_S \sigma_M(t_M)\sqrt{t_M}\Phi^{-1}(1 - m_\epsilon) \tag{194}$$

$$y_n = \ln\left(\max_{i=1}^{M} F(t_i)\right) + m_S \sigma_m(t_M)\sqrt{t_M}\Phi^{-1}(1 - m_\epsilon) \tag{195}$$

where $\sigma_M(t_M)$ is the market volatility for an ATMF call option with expiry t_M.

If the mesher concentration m_c is not given, a uniform grid $y_{i i=1,\ldots,n}$ will be built. Otherwise a grid concentrated around and contaning $\ln(S)$ as a grid point will be contructed following the approach described in [37], 7.11.3., i.e. using an $\sinh(\cdot)$-transformation for the uniform grid points.

To each state variable grid point y_i we associate the width

$$\Delta y_i = \frac{y_{i+1} - y_{i-1}}{2} \tag{196}$$

where we set $y_0 = y_1 - (y_2 - y_1)/2$ and $y_{n+1} = y_n + (y_n - y_{n-1})/2$. The initial Dirac condition $\delta(y - y(0))$ is numerically approximated as

$$p_j = \alpha/\Delta y_j \tag{197}$$

$$p_{j+1} = (1 - \alpha)/\Delta y_{j+1} \tag{198}$$

where j is the unique index for which $y_j \leq y(0) < y_{j+1}$ and $\alpha = (y_{j+1} - y(0))/(y_{j+1} - y_j)$ (notice that $1 \leq j < n - 1$). This ensures

$$\sum_{i=1}^{n} p_i \Delta y_i = 1 \tag{199}$$

The Fokker-Planck PDE is solved numerically using the Crank-Nicholson scheme. The first d steps are evolved using an implicit Euler scheme to account for the singular initial condition, with $d = 5$. The calibration proceeds as follows (also cf. section 4.2 of [25]):

1. Set $i = 1$.
2. Evolve the PDE for the discretised probability p from t_{i-1} to t_i on the refined time grid t'_j, i.e. stepping over all $t_{i-1} \leq t'_j \leq t_i$ using preliminary parameters h_0^* and σ^* for the piecewise model functions h_0 and σ on the interval $[t_{i-1}, t_i)$ (where we set $t_0 := 0$). For $i > 1$ start with the parameters found for $i - 1$. For $i = 1$ start with parameters that are calculated for $p = 0$ as a start value.
3. Compute the model prices for a defaultable zero bond with maturity t_i as

$$P_D^{mdl}(0, t_i) = \sum_{i=1}^{n} p_i \Delta y_i$$

and for an equity option with expiry t_i and ATMF strike K [12]

$$C^{mdl}(K, t_i) = \sum_{i \text{ s.t. } y_i > \ln(K)} p_i \Delta y_i (e^{y_i} - K)$$

4. Minimize $T(h_0^*, \sigma^*)$ using a Levenberg-Marquardt optimizer by repeating step 2 until the optimizer has converged to a solution. The minimization is either done (chosen via the pricing configuration parameter `Bootstrap.Mode`)

   - Simultaneously: The target function is defined as

     $$T(h_0^*, \sigma^*) = \left(\frac{P^{mdl} - P^{mkt}}{P^{mkt}} \right)^2 + \left(\frac{C^{mdl} - C^{mkt}}{C^{mkt}} \right)^2$$

     and minimized in two variables simultaneously.

   - Alternating: The target function

     $$T(h_0^*) = \left(\frac{P^{mdl} - P^{mkt}}{P^{mkt}} \right)^2$$

     is minimized with fixed σ^*. The found solution h^* for h_0 is kept and the target function

     $$T(\sigma^*) = \left(\frac{C^{mdl} - C^{mkt}}{C^{mkt}} \right)^2$$

     is minimized with this fixed h^*. The found solution σ^* for σ is kept and the two optimizations in h^* and σ^* and repeated until the change in both solutions is below a threshold.

5. Set the piecewise flat model functions h_0 and σ on $[t_{i-1}, t_i)$ to the solution found in the optimisation for step i.

6. Increase i by 1 and go to 2 if $i < m$, otherwise stop.

After the procedure above has finished we have calibrated model functions $h_0(t)$ and $\sigma(t)$ so that market defaultable zero bonds and ATMF equity optinos are repriced by the model on the calibration grid $t_1, \ldots, t_m$. For $t > t_m$ we extrapolate the model functions flat (notice this is never used in the context of convertible bond pricing due to the construction of the calibration grid, see the start of this section).

Numerical Implementation of the Pricing

For the pricing of the convertible bond, we build a set of event times

$$t_1, \ldots, t_m \tag{200}$$

associated to pricing events

- underlying bond flow payment
- issuer call date

[12] for the smallest i s.t. $y_i > \ln(K)$ we replace Δy_i with $(y_i - \ln(K)) + (y_{i+1} - y_i)/2$ to account for the fact that $\ln(K)$ in general does not lie on the midpoint between $y_{i-1} \leq \ln(K) < y_i$.

- investor put date
- voluntary conversion date
- mandatory conversion date
- contingent conversion trigger observation date
- conversion ratio reset (due to a conversion reset or dividend protection with conversion ratio adjustment)
- dividend pass through (due to dividend protection with pass through)

The set of event times is enriched to ensure a minimum number of grid points per year (taken from parameter `Pricing.TimeStepsPerYear` in the pricing engine configuration) for the numerical PDE solution, similar to what is done for the calibration step:

$$t_1, \dots, t_m \to t'_1, \dots, t'_M \tag{201}$$

The refined times t'_i are also used to approximate American call / puts and conversion rights, i.e. it is assumed that on each such grid point the respective right can be exercised as an approximation to daily exercises of these options.

The state variable $y = \ln(S)$ is discretised on a uniform grid

$$y_1, \dots, y_n \tag{202}$$

where the number of grid points is taken from the pricing configuration parameter `Pricing.StateGridPoints`. The boundaries are determined in a similar way to the calibration step, i.e.

$$y_1 = \ln\left(\min_{i=1}^{M} F(t_i)\right) - m_S \sigma_M(t_M)\sqrt{t_M}\Phi^{-1}(1 - m_\epsilon) \tag{203}$$

$$y_n = \ln\left(\max_{i=1}^{M} F(t_i)\right) + m_S \sigma_m(t_M)\sqrt{t_M}\Phi^{-1}(1 - m_\epsilon) \tag{204}$$

with m_ϵ taken from `Pricing.MesherEpsilon`, m_S from `Pricing.MesherScaling`.

The PDE is solved using a Crank-Nicholson scheme. We use the following vectors during the rollback. The vector v is initialised with 0 at $t = t_m$. The vector w is left uninitialised for the time being:

- $(v_i)_{i=1,\dots,n}$: the value of the convertible bond at y_i
- $(w_i)_{i=1,\dots,n}$: the value of the convertible bond at y_i assuming no conversion is possible during conversion periods with contingent conversion triggers observed at the start of these periods

The vector w is used to model contingent conversion rights where the trigger is observed at the start of a conversion period. In this case, both v and w need to be

rolled back and on the observation date it is decided how to update the value vector v, see below for details.

In the presence of either conversion reset events or dividend protection events with conversion ratio adjustments, instead of v and w we need several parallel versions of v and w for a discretised set of conversion ratios $cr_1, \ldots, cr_K$. The conversion ratios are discretised by setting

$$cr_k = CR \cdot m_k \tag{205}$$

where the multipliers m_k for $k = 1, \ldots, K$ are taken from the pricing engine parameter `Pricing.ConversionRatioDiscretisationGrid` and CR is the initial conversion ratio from the term sheet. We then have to deal with $2K$ vectors instead of only 2:

- $(v_i^k)_{i=1,\ldots,n}^{k=1,\ldots,K}$
- $(w_i^k)_{i=1,\ldots,n}^{k=1,\ldots,K}$

As before, the vectors v^k are initialised with 0 while the w^k are left uninitialised during periods without contingent conversion rights. To ease notation we will write v_i^k for v_i in the following if no discretization of the conversion rate is required, i.e. we set $k = K = 1$ in this case.

The pricing algorithm proceeds as follows:

1. Set $i := M$
2. If on t_i there is a conversion right contingent on the observation of a trigger at an earlier $t_{i'}$, $i' < i$ and w^k is not initialised, initialise w^k with v^k (for $k = 1, \ldots, K$)
3. Update v_j^k according to voluntary (contingent) conversion rights:
   (a) compute the exercise value $E_j = cr_k e^{y_j} N_t / N_0$
   (b) if there is a contingent conversion trigger check on t_i:
      i. if the check is on the start of period check for future contingent converion rights: if the conversion right is not triggered, i.e.
      $$cr_k e^{y_j} \leq B$$
      set v_j^k to w_j^k (future conversion is not possible), otherwise do not update v_j^k (future conversion is possible). Uninitialise all w^k's.
      ii. if the check is a spot check and the conversion right is triggered, i.e.
      $$cr_k e^{y_j} > B$$
      set v_j^k to $\max(v_j^k, E_j)$.
   (c) if there is no contingent conversion trigger check on t_i, update v_j^k to $\max(v_j^k, E_j)$
4. If on t_i there is a conversion ratio reset (due to a conversion reset feature or a dividend protection or a reset of the conversion ratio to a fixed value)
   (a) for all $k = 1, \ldots, K$, $j = 1, \ldots, n$:

(b) interpolate v_j^k linearly from the v_j^κ on the adjusted conversion ratio $cr' = cr'(cr_k, y_j)$ which is always computable as a function of the assumed conversion ratio for v^k and the log equity price y_j for the involved reset and dividend protection features.

(c) interpolate w_j^k from w_j^κ analoguously, if w is initialised

5. If no conversion resets occur on t_i or prior to t_i, collapse the v^k to one v by interpolating the v from the v^k on the initial conversion ratio, and do the same with w^k (if initialised)

6. If there is a mandatory conversion right on t_i set v_j^k (and w_j^k if initialised) to the mandatory conversion payoff, which is a function of y_j.

7. If there is an issuer call right on t_i and no conversion (voluntary or mandatory) has been exercised in the above steps on the same t_i, update

$$v_j^k \to \min(\max(c, f), v_j^k)$$

where f is the forced conversion value

$$f = cr_k e^{y_j} N_t / N_0$$

and c is the amount to be paid by the issuer in case of a call. If the call is soft, only update v_j^k if the soft call is triggered, i.e.

$$e^{y_j} > TN_0/cr_k$$

If w is initialised, update w analogously. Notice that a forced coversion can be exercised regardless of a contingent conversion clause active.

8. If there is an investor put on t_i, update

$$v_j^k \to \max(v_j^k, p)$$

where p is the put price to be paid by the issuer in case of a put exercise by the investor. Notice that if a call was exercised before, the put overrides the call. Also, a put overrides a conversion (voluntary or mandatory) on the same date, because the investor will exercise the put or conversion whatever is more valuable, and in case of a mandatory conversion and a put on the same date (unlikely in a real term sheet) we favor the put over the mandatory conversion.

9. If there is a dividend protection pass through on t_i add the relevant amount to v_j^k (and w_j^k if initialised).

10. If a bond cashflow is paid on t_i, add the amount to v_j^k (and w_j^k if initialised)

11. Roll back v_j^k (and w_j^k if initialised) from t_i to t_{i-1}. If $i > 1$ go to step 2.

After the algorithm has finished, the NPV of the convertible is interpolated from v_j.

Additional pricing engine results

The following additional results are provided by the pricer:

Results concerning the processing of events on the FD grid:

- time: the time associated to the FD grid point
- date: an event date associated to the time
- notional: the current notional of the underlying bond
- accrual: the current accrual of the underlying bond
- flow: bond flow payments
- call: issuer call events:
  - `@1.0` hard call with price 1.0
  - `@1.0 s@1.3` soft call with price 1.0 and trigger 1.3
- put: investor put events:
  - `@1.0` put with price 1.0
- conversion: conversion event:
  - `@0.03` voluntary conversion with conversion ratio 0.03
  - `@0.03 c@1.3` contingent conversion with conversion ratio 0.03 and coco barrier 1.3
  - `@0.03 c@1.3b` as before, but check is on the next date in the past without being marked with `b`
  - `peps(0.03,0.04)`: mandatory conversion with PEPS barriers 0.03 and 0.04
- CR_reset: conversion ratio reset event:
  - `0.8@0.9/CP0` conversion ratio reset with gearing 0.8 and threshold 0.9, reference price is the initial conversion price
  - `0.8@0.9/CPT` as before, but reference price is the current conversion price
  - `DP(i,0.23)@0.17` dividend protection conversion ratio reset, relevant period starts at time index $i+1$, dividend yield over relevant period (without historical dividends) is 0.23, threshold is 0.17
- div_passth: dividend passthroughs:
  - `@0.17`: dividend pass-through event with threshhold 0.17
- curr_cr: the current conversion ratio associated to the time index, if there are conversion ratio reset events or dividend protection events with conversion ratio reset active, the number is displayed with the suffix `s` for stochastic (in which case several PDE planes for different discretized conversion ratios are rolled back)
- fxConv: the fx conversion factor from Equity to Bond currency for the time index.
- eq_fwd: the equity forward price at the time index (value is output just for info purposes)

- div_amt: the expected dividend amount relevant for dividend pass through or conversion rate adjustments based on the dividend protection feature (calculated using the equity forward price, value is output just for info purposes)
- conv_val: the expected conversion value at the time index based on the equity forward and current conversion ratio (excluding stochastic adjustments, value is output just for info purposes)
- conv_prc: the expected conversion price at the time index based on the equity forward and current conversion ratio (excluding stochastic adjustments, value is output just for info purposes)

Results concerning the pricing:

- BondFloor: the bond floor value
- trade.tMax: the maximum time relevant for the trade
- market.discountRate(tMax): the benchmark curve discount rate at tMax
- market.creditSpread(tMax): the hazard rate at tMax, for exchangeables this is the hazard rate associated to the equity credit risk
- market.exchangeableBondSpread(tMax): the hazard rate at tMax associated to the bond credit risk (only filled for exchangeables)
- market.discountinngSpread: the additional (security) spread applied to discounting
- market.recoveryRate: the recovery rate applied in the model (in case of a bond default)
- market.equitySpot: the equity spot
- market.equityForward(tMax): the equity forward at tMax
- market.equityVolatility(tMax): the equity market volatility at tMax
- model.fdGridSize: the number of grid points for the FD solve (in time direction)
- model.eta: the model parameter eta (equity default loss ratio)
- model.p: the model parameter p (credit-equity linkage)
- model.calibrationTimes: the times on which the model is calibrated
- model.h0: the calibrated model parameter h_0 on the calibration time grid
- model.sigma: the calibrated model parameter σ on the calibration time grid
- conversionIndicator: the pv of a hypothetical instrument paying 1 when the conversion is exercised

Results concerning the processing of events before or on the evaluation date:

- historicEvents.initialConversionRatio: The initial conversion ratio of the bond.
- historicEvents.crReset: Data concerning historic conversion ratio resets:
  - crReset_DATE_S: the share price on the reset date

  - crReset_DATE_threshold: the threshold
  - crReset_DATE_referenceCP: the reference conversion price
  - crReset_DATE_gearing: the gearing
  - crReset_DATE_floor: the floor
  - crReset_DATE_globalFloor: the global floor
  - crReset_DATE_currentCr: the conversion ratio before the reset event
  - crReset_DATE_adjustedCr: the conversion ratio after the reset event
- historicEvents.crReset_DP: Data concerning historic dividend protection conversion ratio adjustments:
  - crReset_DP_DATE_div_DATE1_DATE2: the dividends for the adjustment period date1/2
  - crReset_DP_DATE_S: the share price on the reset ddate
  - crReset_DP_DATE_threshold: the threshold
  - crReset_DP_DATE_currentCr: the conversion ratio before the reset event
  - crReset_DP_DATE_adjustedCr: the conversion ratio after the reset event
- historicEvents.coco: Data concerning contingent conversion with start of period observation:
  - coco_DATE_S: the share price on the observation date
  - coco_DATE_cocoBarrier: the barrier
  - coco_DATE_currentCr: the conversion ratio on the observation date
  - coco_DATE_triggered: whether the barrier is triggered (i.e. conversion is allowed)
- historicEvents.accruedDividends_DATE1_DATE2: dividends between the last dividend protection date before the evaluation date and the evaluation date, which enter the next (future) dividend reset event or pass through

5.10.4 ASCOT

ASCOT's are priced using a naive intrinsic model described in [32] 8.6.1.

5.10.5 Collateral Bond Obligaton CBO

We consider an n tranche CBO, or Cashflow CDO. The underlying assets consist of a portfolio of corporate bonds or loans with either amortising or bullet structures. The portfolio can contain fixed or floating rate obligations. Maturities cover a range and do need not coincide. We assume that hazard rate and/or security spread data is available and provided externally.

The deal is assumed to be structured as a cashflow securitisation. Interest and Notional repayments are directed in an order of priority first to the senior note holder. We assume the available pool for Notional repayments consists of scheduled bond

notional repayments and recovery amounts. The pool available for interest payments consists of coupons received on the portfolio during the payment period in question.

Class N notes or equity receive the excess pool coupon available after other items in the interest waterfall are discharged.

A typical Interest and Notional waterfall is given in the following.

Interest Waterfall:

1. Senior Fees
2. Interest due on Senior notes
3. Redemption of Senior notes if over-collateralisation or interest coverage tests not met (sufficient to ensure tests are met)
4. Interest due on next most Senior notes
5. Redemption of next most Senior notes if over-collateralisation or interest coverage tests not met (sufficient to ensure tests are met)
6. $\vdots$
7. Subordinated Fees
8. Residual split between management incentive fee and Equity notes

We explicitly write down the pricing formula for each of the CBO notes. First some notation:

- t_i are the scheduled payment dates of the trade,
- $C_k(t_i)$ are the coupons paid on the Class k notes on the ith payment date
- $N_k(t_i)$ are the outstanding notionals.
- $R_k(t_i)$ are the notional redemptions payable on the ith payment date. Note $N_k(t_i) = N_k(t_{i-1}) - R_k(t_i)$.
- $D(t, t_i)$ are the discount factors with respect to the evaluation date t.
- Π_k is the present values of the notes which we seek

When we take expectations we assume that time to default and interest rates are independent and so all payments are multiplied by the Banks risk free discount factor to the payment date and the symbol $\mathbb{E}$ is reserved for expectations under the Credit Measure.

Using these conventions we can write the present values as

$$\Pi_k = \sum_{i=1}^{n} \mathbb{E}(C_k(t_i))D(t, t_i) + \sum_{i=1}^{n} \mathbb{E}(R_k(t_i))D(t, t_i). \tag{206}$$

We will assume in the following that we know the marginal probability of default of each underlying asset to any desired date and that we use a standard copula framework (see section 5.6.6) to compute expectations in the presence of joint defaults.

MonteCarlo algorithm

A systematic numerical way of evaluation the time to default of the redemption flows is a MonteCarlo simulation. Correlated default times are generated by drawing correlated Gaussian random variates for each sample path and each underlying bond / loan. Those variates, e.g. y, are transformed into default times exploiting the following relationship:

$$\tau = \frac{ln\left(1 - N\left(y\right)\right)}{h\left(t\right)}$$

The redemption / principal flows together with the interest flows are assigned into a time grid created from the tranche's payment dates. Those collections are allocated subsequently on each MC path in each time step to through the following waterfall structure to the tranches. For simplicity a two tranche case is shown, more tranche are possible.

- Senior fees (paid from interest collections)
- Interest due on senior notes (paid from interest collections)
- Redemption on senior notes, upon coverage test breach (paid from interest collections)
- Interest due on junior notes (paid from interest collections)
- Redemptions on senior notes until paid in full (paid from principal collections)
- Redemptions on junior notes until paid in full (paid from principal collections)
- Junior fees (paid from interest collections)
- Residual payments split between junior note and incentive fee (paid from both collections)

These tranche cashflows can be discounted to calculate the NPV for each tranche in each MC path. Finally, the reported NPV is the average of all path NPVs.

6 Pricing Models

6.1 Bachelier Model

Under the Bachelier model it is assumed the asset price S is normally distributed and follows an Arithmetic Brownian Motion process (Haug, 1997) Ch 1.3.1:

$$dS = \sigma(t)\, dW(t)$$

where $dW(t)$ is a Wiener process.
Then, the Bachelier model gives the following analytical solution for the price at expiry of a call/put option with strike K, on an underlying asset with price S, and normal volatilities σ:

$$\text{Bachelier}(K, S, v, \omega) = v\,\phi\left(\omega\,\frac{S-K}{v}\right) + \omega\,(S-K)\,\Phi\left(\omega\,\frac{S-K}{v}\right)$$

where

- K: strike price
- S: price of the underlying asset
- T: time to expiration in years
- terminal volatility v: $v^2(T) = \int_0^T \sigma^2(t)\,dt$
- $\sigma(t)$: instantaneous volatility of the underlying asset price
- ω: 1 for a call option and -1 for a put option
- $\Phi(\cdot)$: the cumulative standard normal distribution
- $\phi(\cdot)$: the standard normal density function

6.2 Black Model, Shifted Black Model

In Black's model it is assumed that variable F for future times t follows a Geometric Brownian Motion:

$$dF/F = \mu(t) + \sigma(t)\,dW(t)$$

where $\mu(t)$ is the drift and $dW(t)$ is a Wiener process. If F is a forward price or yield, and the Wiener process is assumed in the T-forward measure, then the Geometric Brownian Motion is drift-free, $\mu(t) \equiv 0$.

This leads to the Black76 analytic formula, see (Lichters, Stamm, Gallagher, 2015) Appendix C.1, for the price at expiry of a call or put option on underlying asset F with strike K and log-normal volatilities σ:

$$\text{Black}(K, F, v, \omega) = \omega\,\{F\,\Phi(\omega\,d_+) - K\,\Phi(\omega\,d_-)\}$$

where

- K: strike rate
- F: the underlying asset forward price F
- $\sigma(t)$: the instantaneous log-normal volatility of F
- terminal volatility v: $v^2 = \int_0^T \sigma^2(t)\,dt$
- ω: 1 for call, -1 for put
- $\Phi(\cdot)$: the cumulative standard normal distribution
- $d_+ = \frac{1}{v}\left(\ln\frac{F}{K} + \frac{1}{2}v^2\right)$
- $d_- = d_+ - v$

Shifted Black model

To apply the Black model in low- or negative-rate scenarios, one can assume displaced Geometric Brownian Motion, introducing a displacement parameter $\alpha \geq 0$ into the dynamics

$$d(F + \alpha) = (F + \alpha)\,\sigma(t)\,dW$$

This parameter is the same for the entire volatility surface.

The present value of a call or put (assuming displaced GBM) uses the same analytical solution as the previous section (Black formula), but with forward rate F and strike K amended to include the displacement parameter α:

$$F_{\text{shifted}} = F + \alpha$$
$$K_{\text{shifted}} = K + \alpha$$

6.3 Linear Terminal Swap Rate model (LTSR)

CMS-linked cash flows (in CMS Swaps and CMS Cap/Floors) are priced using Terminal Swap Rate Models to replicate the payoff with quoted market Swaptions, i.e.

$$\begin{aligned} V(0) &= A(0)\ \mathbb{E}(\alpha(S(t))\, f(S(t))) \\ &= A(0)\, \alpha(S(0))\, f(S(0)) + \int_{-\infty}^{S(0)} f''(k)\, p(k)\, dk + \int_{S(0)}^{\infty} f''(k)\, c(k)\, dk \end{aligned}$$

where

- $V(0)$: the NPV at time zero of a CMS-linked cash flow
- $\mathbb{E}(\dots)$: the expectation in the physical annuity measure
- $S(t)$: the relevant Swap rate observed at fixing time t
- $A(t)$: the NPV of the annuity (w.r.t. physical settlement) at fixing time t
- $P(t,T)$: the discount factor between fixing time t and payment time T
- $\alpha(s) = \mathbb{E}(P(t,T)/A(t)|S(t) = s)$: the annuity mapping function
- $f(S(t))$: the payoff depending on $S(t)$ and paid at time T
- $p(k)$: the NPV of a put (receiver Swaption with physical settlement), struck at strike k
- $c(k)$: the NPV of a call (payer Swaption with physical settlement), struck at strike k

See (Andersen & Piterbarg, 2010), Ch. 16.3 and Ch. 16.6.

The LTSR is specified by the annuity mapping function $\alpha(s) = a\, s + b$, where the coefficient b is determined by a "no arbitrage" condition and a is linked to a one-factor model mean reversion parameter κ, see (Andersen & Piterbarg, 2010), formulas 16.27 and 16.28, for details. It supports normal, log-normal, and shifted log-normal volatilities. Besides the mean reversion, the lower and upper integrations bounds are the parameters of this model.

For log-normal volatilities, the Hagan model can also be used. Hagan model parameters are Mean Reversion and Yield Curve Model. The Yield Curve Model parameter can take the following values: Standard, ExactYield, ParallelShifts, NonParallelShifts. Each of these models corresponds to a certain specification of the annuity mapping function $\alpha(s)$ [40].

6.4 Bivariate swap rate model BrigoMercurio

CMS Spread Options with payoff $\max(\varphi(S_1(t) - S_2(t) - K), 0)$ are priced using the the model [31] Ch 13.6.2, optionally with partial smile adjustment [29]. This model supports log-normal swap rate dynamics and has been extended to support shifted log-normal and normal dynamics as well [34].

The model is given by:

$$\begin{aligned} dS_1 &= \mu_1 (S_1 + d_1)\, dt + \sigma_1 (S_1 + d_1)\, dW_1 \\ dS_2 &= \mu_2 (S_2 + d_2)\, dt + \sigma_2 (S_2 + d_2)\, dW_2 \\ dW_1\, dW_2 &= \rho\, dt \end{aligned}$$

in the shifted lognormal case ($d_1 = d_2 = 0$ in the lognormal case), and by

$$\begin{aligned} dS_1 &= \mu_1 dt + \sigma_1\, dW_1 \\ dS_2 &= \mu_2 dt + \sigma_2\, dW_2 \\ dW_1\, dW_2 &= \rho\, dt \end{aligned}$$

in the normal case. All equations are in the T-forward measure.

Here:

- t: fixing Time
- T: payment Time
- S_1, S_2: Swap Rates
- K: strike
- φ: +1 for a call, -1 for a put
- μ_1, μ_2: implied drifts, see below
- σ_1, σ_2: swap rate volatilities
- ρ: instantaneous correlation

The drifts are determined as deterministic, constant values such that the T-forward expectation matches the convexity-adjusted swap rate with identical fixing and payment time as the CMS Spread payoff. The convexity adjustment for the single swap rates are computed using one of the models in Section 6.3.

The swap rate volatilities are either taken as the market ATM volatilities of the swap rates with the option expiry equal to the fixing time of the CMS Spread payoff, or (optionally) as smile volatilities with strikes $S_2 + K$ (for S_1) and $S_1 - K$ (for S_2), see [29] for the rationale of the latter.

The instantaneous correlation is an external input to the model, which can be implied from quoted CMS Spread option premiums.

The (shifted) lognormal model allows for a semi-analytical solution involving a one-dimensional numerical integration which can be carried out using a Gauss-Hermite integration scheme, see [31] Ch 13.6.2 for the details.

The normal model can be solved using the Bachelier option pricing formula, see [34]. In this special case the NPV ν is given by:

$$P(0,T)\mathcal{B}(\phi, K, \mu, \sigma\sqrt{t}) \tag{207}$$

Here, $\mathcal{B}$ denotes the Bachelier option pricing formula with strike K, forward μ, standard deviation $\sigma\sqrt{t}$ and $\phi = 1$ for a call, -1 for a put. The forward is given by

$$\mu = (S_1(0) + \mu_1 t) - (S_2(0) + \mu_2 t); \tag{208}$$

and the variance is given by

$$\sigma^2 = \sigma_1^2 t + \sigma_2^2 t - 2\rho\sigma_1\sigma_2 t \tag{209}$$

6.5 One-Factor Linear Gauss Markov model (LGM)

The starting point for the one-factor LGM model is the one-factor Hull-White (HW) model with time-dependent parameters in the bank account measure.

$$dr_t = (\theta_t - \lambda_t r_t)\, dt + \sigma_t\, dW_t$$

where:

- θ_t: long-term mean interest rate
- λ_t: mean reversion parameter
- r_t: short interest rate
- σ_t: volatility of the short rate
- dW_t: a Wiener process

The LGM model is closely related to the HW model, and it can e.g. be obtained from the HW model by means of a change of variables and change of measure. In the LGM, both numeraire and zero bond price are closed-form functions

$$N(t) = \frac{1}{P(0,t)} \exp\left\{ H_t\, z_t + \frac{1}{2} H_t^2\, \zeta_t \right\}$$
$$P(t,T,z_t) = \frac{P(0,T)}{P(0,t)} \exp\left\{ -(H_T - H_t)\, z_t - \frac{1}{2}\left(H_T^2 - H_t^2\right) \zeta_t \right\}.$$

of a single Gaussian random variable z_t,

$$dz_t = \alpha_t\, dW_t.$$

The LGM parameters H_t and ζ_t can be related to the HW model's mean reversion speed λ_t and volatility σ_t above. The model's volatility functions are calibrated to Swaptions, choosing a limited range of Swaption expiries and underlying Swap maturities from the ATM surface or adapted to the deal strike. For details refer to [46] Ch. 11.1.

Pricing with LGM is done, depending on product type, either

- semi-analytically using the closed form expressions for numeraire and zero bond above
- using a backward induction approach on a one-dimensional LGM lattice, or
- using Monte Carlo simulation.

6.6 Barone-Adesi and Whaley Model

In the Barone-Adesi and Whaley model, the price of an American option is subdivided into a European option component (priced with Black76) and an early exercise premium component. Summed up, the prices of these two components give the American option price.

The early exercise premium is approximated with a free boundary value problem for an ordinary differential equation, which in turn, is solved numerically by quadratic approximation.

The starting point for the approximation is the Black-Scholes partial differential equation for the value of an FX Option with price V:

$$\frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + (r_d - r_f) S \frac{\partial V}{\partial S} - r_d V + \frac{\partial V}{\partial t} = 0$$

where:

- V: the option price
- S: the underlying spot FX rate between domestic and foreign currencies
- σ: the volatility of the FX rate
- r_d: the domestic interest rate
- r_f: the foreign interest rate

For American options, there is no known analytic solution to the differential equation above. Barone-Adesi and Whaley decomposes the American option price into the European price and the early exercise premium:

$$C_{Amer}(S,T) = C_{Euro}(S,T) + E_c(S,T)$$

where:

- $C_{Amer}(S,T)$: the price of an American FX call option on underlying FX rate S at expiry T
- $C_{Euro}(S,T)$: the price of a European FX call option on underlying FX rate S at expiry T
- $E_c(S,T)$: the early exercise premium of the American FX call option

Using that $E_c(S,T)$ also satisfies the Black-Scholes partial differential equation, and removing terms involving $\partial V/\partial t$ that are negligible, it is found that:

$$C_{Amer}(S,T) = \begin{cases} C_{Euro}(S,T) + A_2 \left(\frac{S}{S^*}\right)^{q_2} & ; \quad S < S^* \\ S - K & ; \quad S \geq S^* \end{cases}$$

where:

- $A_2 = \frac{S^*}{q_2}\left(1 - e^{-r_f(T-t)}\right)\,\Phi(d_1\,S^*)$
- $q_2 = \frac{1}{2}\left(-(\beta-1) + \sqrt{(\beta-1)^2 + 4\alpha/h}\right)$
- $\alpha = \frac{2\,r_d}{\sigma^2}$
- $\beta = 2\frac{r_d - r_f}{\sigma^2}$
- $h(T) = 1 - e^{-r_d(T-t)}$
- $d_1(S) = \frac{1}{\sigma\sqrt{T-t}}\left(\ln\frac{S}{K} + \left(r_d - r_f + \frac{\sigma^2}{2}\right)(T-t)\right)$
- S^*: the FX rate below which the option should be exercised
- K: the strike FX rate
- S^* solves: $S^* - K = C_{Euro}(S^*, T) + \frac{S^*}{q_2}\left(1 - e^{-r_f(T-t)}\right)\Phi(d_1\,S^*)$

For more details refer to the original Barone-Adesi & Whaley paper [28].

7 Curve Building

Curve construction is comprised of the following fundamental steps:

- Determine the financial instruments that will provide the best indication of the curve rates for each section of the curve.
- Determine the discount curve to apply to the quotes given by the selected financial instruments.
- Use a numerical method to compute fair rates of constituent financial instruments given "'discount" and "forward" curves
- Use bootstrapping techniques (iterative procedures) to determine the discount or forward curve that yields the implied fair rates (see above) consistent with the given market quotes
- The bootstrap also involves the choice of interpolation methods (linear, log-linear, cubic spline, financial cubic spline, etc.) and interpoaltion domains (discount factors, zero rates, forward rates) to determine curve rates for maturities other than those of the given quotes.

7.1 Interest Rates

The yield curve construction described here assumes perfect CSAs with cash collateral in a single unique currency (CSA currency), daily margining and vanishing thresholds and minimum transfer amounts. Any deviation from these assumptions requires the calculation of value adjustments to take CSA imperfections into account.

Each CSA currency then determines a coherent set of yield curves that allow consistent pricing of single and cross currency derivatives.

Four main types of IR curves are built for such a curve set:

- Overnight Index Swap curves (OIS curves). These curves are constructed from Overnight Deposit, Futures and OIS quotes, and they are used in Overnight Index Swap pricing. The CSA currency OIS curve moreover forms the "domestic" discount curve of the set, i.e. it is used to discount cash flows in CSA currency. OIS markets are available in a limited number of economies including EUR, USD, GBP, CHF, JPY, AUD, CAD, SGD, HKD, with the first four being the most developed with longest maturity swaps.
- IBOR index curves for index tenors longer than overnight (1M, 3M, 6M, etc.). These curves are constructed from Deposit, Forward Rate Agreement, Interest Rate Swap quotes, and/or Single Currency Basis Swap quotes, and they are used in Interest Rate Swap and Basis Swap pricing.
- "Foreign" currency discount curves, for discounting derivative cashflows in a currency different from the CSA Currency; these curves are implied from FX Forwards and Cross-Currency (Basis) Swaps where available
- LIBOR Fallback curves. Since the Libor cessation on 30 June 2023, some LIBOR curves have ceased. The fallback for the ceased curves is the OIS RFR index of the respective LIBOR index plus a defined fallback spread specific to the index.

As a tangible example, consider the CSA currency USD, and the following curve building sequence in ORE [48].

Step 1: The USD OIS curve can be either the USD-FedFunds curve or the USD-SOFR curve

The USD-FedFunds curve building would typically take Overnight Deposit and OIS quotes (Swaps with a Overnight tenor) into account

Term	Instrument
1D	MM/RATE/USD/0D/1D
1M	IR_SWAP/RATE/USD/2D/1D/1M
3M	IR_SWAP/RATE/USD/2D/1D/3M
6M	IR_SWAP/RATE/USD/2D/1D/6M
9M	IR_SWAP/RATE/USD/2D/1D/9M
1Y	IR_SWAP/RATE/USD/2D/1D/1Y
15M	IR_SWAP/RATE/USD/2D/1D/1Y3M
18M	IR_SWAP/RATE/USD/2D/1D/1Y6M
21M	IR_SWAP/RATE/USD/2D/1D/1Y9M
2Y	IR_SWAP/RATE/USD/2D/1D/2Y
...	...
50Y	IR_SWAP/RATE/USD/2D/1D/50Y

The USD-SOFR curve building would typically take Overnight Deposit, Futures and OIS quotes into account

Term	Instrument
1D	MM/RATE/USD/0D/1D
1M to 2Y	OI_FUTURE/PRICE/USD/YYYY-MM/XCME:SRA/3M
2Y	IR_SWAP/RATE/USD/2D/1D/2Y
3Y	IR_SWAP/RATE/USD/2D/1D/3Y
4Y	IR_SWAP/RATE/USD/2D/1D/4Y
...	...
50Y	IR_SWAP/RATE/USD/2D/1D/50Y

Step 2: Build the IBOR curve. Before LIBOR cessation step 2 would be to build a USD-LIBOR-3M index curve where the USD OIS curve built in step 1 would be used for discounting. This is no longer required for USD. But as an example of an IBOR curve build in step 2, we use the EUR-EURIBOR-3M index curve assuming the EUR OIS was built in step 1. The EUR-EURIBOR-3M index curve is typically built using Deposits, Futures, and Interest Rate Swaps, where the EUR OIS curve is used for discounting. The curve building yields the index projection curve as a result.

Term	Instrument
3M	MM/RATE/EUR/2D/3M
3M to 2Y	OI_FUTURE/PRICE/EUR/YYYY-MM/XICE:FEI/3M
2Y	IR_SWAP/RATE/EUR/2D/3M/2Y
3Y	IR_SWAP/RATE/EUR/2D/3M/3Y
4Y	IR_SWAP/RATE/EUR/2D/3M/4Y
5Y	IR_SWAP/RATE/EUR/2D/3M/5Y
...	...
60Y	IR_SWAP/RATE/EUR/2D/3M/60Y

Similarly, we would build index curves for other tenors and currencies such as EUR-EURIBOR-6M, AUD-BBSW-6M, etc. using the respective currency's OIS curve (EUR Eonia/Ester, AUD Aonia) for discounting.

Step 3: Build the "foreign" currency discount curve – for example for cash flows in EUR – from FX Spot, FX Forward, and Cross Currency Basis Swap quotes:

Term	Instrument
FXSPOT	FXSPOT/EUR/USD
FXON	FXON/EUR/USD
FXTN	FXTN/EUR/USD
1W	FXFWD/RATE/EUR/USD/1W
2W	FXFWD/RATE/EUR/USD/2W
3W	FXFWD/RATE/EUR/USD/3W
1M	FXFWD/RATE/EUR/USD/1M
...	...
1Y	FXFWD/RATE/EUR/USD/1Y
15M	FXFWD/RATE/EUR/USD/15M
18M	FXFWD/RATE/EUR/USD/18M
21M	FXFWD/RATE/EUR/USD/21M
2Y	CC_BASIS_SWAP/BASIS_SPREAD/USD/1D/EUR/1D/2Y
3Y	CC_BASIS_SWAP/BASIS_SPREAD/USD/1D/EUR/1D/3Y
...	...
30Y	CC_BASIS_SWAP/BASIS_SPREAD/USD/1D/EUR/1D/30Y
40Y	CC_BASIS_SWAP/BASIS_SPREAD/USD/1D/EUR/1D/40Y
50Y	CC_BASIS_SWAP/BASIS_SPREAD/USD/1D/EUR/1D/50Y

This bootstrap takes the USD-SOFR and EUR-ESTER index curves as an input and yields the EUR discount curve ("EUR-IN-USD") that needs to be applied under USD collateral.

A similar construction is done for any other non-CSA currency discount curve ("GBP-IN-USD", "TRY-IN-USD", "CNH-IN-USD" etc). Cross Currency Basis Swaps are used where available. In some economies such as TRY or CNH, we might use fixed-float Cross Currency Swap quotes instead where floating rate legs in USD are exchanged for fixed rate legs in the foreign currency. Or the curve could be built from FX quotes only if discount factors are required for moderate times to maturity only.

7.2 Foreign Exchange

FX forward rates are calculated from the FX spot rate, and discount curves for the two currencies in question. That is, the OIS curve for the CSA currency and the Foreign Currency Discount curve for any other currency.

$$F(t,T) = X(t)\frac{P^A(t,T)}{P^B(t,T)}$$

Where:

- $F(t,T)$: the forward projected FX rate between currency A and B for maturity T at time t
- $X(t)$: the FX spot rate between currency A and currency B at time t
- $P^A(t,T)$: the discount factor for currency A from time T to time t
- $P^B(t,T)$: the discount factor for currency B from time T to time t

Since the Foreign Currency Discount curves are built from quoted FX Forwards and Cross Currency Basis Swaps, the implied FX Forward above is consistent with the market by construction.

7.3 Inflation

There are two types of inflation curves:

- Zero-Coupon Inflation Index curves, built from CPI Swap (or Zero Coupon Inflation Swap) quotes, used to price CPI-linked instruments
- Year-on-Year Inflation Index curves, built from Year-on-Year Inflation Swap quotes, used to price instruments linked to YoY inflation indices

Zero-Coupon Inflation Index curve interpolate the zero inflation rate linearly, Year-on-Year Inflation Index curves interpolate the Year-on-Year rate linearly. Before the first pillar extrapolation is usually flat, if a base rate is specified the first curve segment is interpolated linearly between this rate and the first market quote. Extrapolation (if enabled) beyond the last curve pillar is linear.

A multiplicative seasonality adjustment can be applied to projected zero inflation rates. The adjusted zero rate r_a at a date d is given by

$$r_a = (1 + r_u)f - 1 \tag{210}$$

where r_u is the unadjusted rate and f is the adjustment factor

$$f = \left(\frac{s(d-l)}{s(d_b)} \right)^{1/t} \tag{211}$$

where t is the time from the curve base date d_b to the date d minus the observation lag l and $s(\cdot)$ denotes the externally given seasonality factor for a specific date. The following relation between the unadjusted and the adjusted zero rates hold, for any given fixing b at the base date:

$$b(1 + r_u)^t \left(\frac{s(d-l)}{s(d_b)} \right) = b(1 + r_a)^t \tag{212}$$

7.4 Equity

Equity forward price curves can be built in two ways:

- From two externally provided curves, a forecasting IR curve and dividend yield term structure; no bootstrap is required in this case
- Alternatively, from a forecasting IR curve and market equity forward/futures price quotes; in this case the dividend yield curve is bootstrapped from the provided quotes

In both cases the equity spot price is required to start the curve building at the short end.

By default, log-linear interpolation in the discount factor domain is used (i.e. piecewise flat forward interpolation) for all curves, though further schemes are available. When extrapolation is enabled, extrapolation is flat.

If the dividend curve is bootstrapped from forward/futures price quotes, it interpolates the implied dividends on the pillars defined by the forward/future expiries.

7.5 Credit

There are two main types of default probability curves:

- Default probability curves bootstrapped from quoted CDS instruments (recovery rate and running CDS spreads by available maturity).
- Default probability curves obtained directly from hazard rates (loss given default and externally estimated hazard rates).

Curve construction methodology applies the following key elements:

- Quoted fixed recovery rates or LGDs are required for each reference entity
- The OIS curve in respective currency is used for discounting in the bootstrapping of default probability curves
- Log linear interpolation of survival probabilities is used
- Flat extrapolation is used when extrapolation is enabled
- Reference entities can be single name CDS or CDS Indices. In the latter case, an Index CDS curve can be bootstrapped from Index CDS quotes in the same way as for a single name.

Default probability curves use log linear interpolation in the survival probability (equivalent to backward flat interpolation of the instantaneous hazard rate).

7.6 Commodity

Commodity forward curves are built directly from market commodity forward quotes for each respective commodity.

Curve construction methodology applies the following key elements:

- Quoted commodity spot price
- Commodity forward price quotes as far as available
- Bootstrapping is not applied as the commodity forward quotes are used directly
- Linear interpolation is used on the forward price quotes
- Flat extrapolation is used when extrapolation is enabled

The forward price is interpolated and extrapolated linearly by default, log-linear interpolation is also available.

7.7 Volatility Structures

Spacing and frequency of strikes, tenors, and expiries forming volatility curves and cubes depend on currency and underlying asset.

Interpolation is linear and depends on the product type and volatility structure, detailed in the following subsections.

7.7.1 Interest Rates - Cap/Floor

For each currency, there are cap/floor volatilities for just one index tenor. For each index tenor / currency, the cap/floor volatility surface is composed of the following two dimensions:

- Absolute strikes
- Option expiries

Note that cap/floor prices may be given for each point on the surface, instead of volatilities. In this case, the volatility for each point is derived from the price.

Caplet/floorlet volatilities are bootstrapped from quoted "flat" caps/floors.

Interpolation is bilinear between the two closest expiries and strikes (i.e. "2D" interpolation)

IR cap/floor volatilities can be normal or log-normal (including shifted log-normal).

7.7.2 Interest Rates - Swaption

The swaption volatility cube has the following 3 dimensions:

- Strikes, quoted as fixed percentage offsets from the ATM forward strike
- Option expiries
- Maturities of the underlying swap

Interpolation starts by linear 2D interpolation on the ATM surface for the two closest expiries and maturities. Then, the skew or offset from ATM is linearly interpolated.

Swaption volatilities can be normal or log-normal (including shifted log-normal).

7.7.3 Foreign Exchange

The FX Volatility surface has the following dimensions:

- Strikes for the ATM, and the Risk Reversal (RR), Vega Weighted Butterfly (VWBF) points. Then the Vanna Volga approach (Castagna, 2006) is used to extend the three strike points to a full smile.
- Option expiries

Linear interpolation is done separately on the ATM, RR, and VWBF term structures, and the resulting points are used to build a Vanna Volga smile for the required volatility.

FX Volatilities are log-normal.

7.7.4 Inflation - Cap/Floor

Cap/floors on both CPI and Year-on-Year inflation have a volatility surface of the following two dimensions:

- Absolute strikes
- Option expiries

Note that cap/floor prices may be given for each point on the surface, instead of volatilities. In this case, the volatility for each point is derived from the price.

Inflation caplet/floorlet volatilities are bootstrapped from quoted "flat" inflation caps/floors.

Interpolation is bilinear between the two closest expiries and strikes (i.e. "2D" interpolation)

Inflation cap/floor volatilities can be normal or log-normal (including shifted log-normal).

7.7.5 Equity

The equity volatility surface has the following 2 dimensions:

- Absolute strikes
- Option expiries

Interpolation is done linearly between the two closest expiries and strikes (i.e. "2D" interpolation)

Equity volatilities are log-normal.

7.7.6 Commodity

The commodity volatility surface has two dimensions:

- Absolute strikes, Relative strikes, or Delta quotes
- Option expiries

Interpolation is done linearly between the two closest expiries and strikes (i.e. "2D" interpolation).

For delta quotes, 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 vs. vol curve is then interpolated (or extrapolated) to give the vol at the (t, s).

Commodity volatilities are log-normal.

8 Libor Fallback

This section describes the handling of coupons referencing a Libor rate after its cessation date.

8.1 Base Line

The publication of fixings for a number of Libor indices will cease at certain dates. In addition we expect that market quotes that we use to build forward curves for these indices will loose liquidity and eventually no longer provided as well. The system contains fallbacks addressing both points to enable seamless processing of coupons without the nececssity to change the trade representation from "Ibor" to the correpsonding "OIS" legs encoding the fallback conventions.

References: [56], [57], [58], [60], [61], [62], [63], [64].

8.2 Standard Libor Coupon Pricing

As long as the sytem's evaluation date lies strictly before an Libor index cessation date, nothing changes, i.e. we use the historical fixings from the usual data feeds and forward curve built from the usual market quotes. As soon as the system's evaluation date lies on or after an Libor index cessation date, the following fallback handling kicks in.

- If the coupon fixing date lies earlier than the index cessation date, the usual "old" published Libor fixing is used.
- If the coupon fixing date lies on or after the index cessation date, a fallback fixing is computed over the relevant interest rate period of the original Libor index with a lookback of 2 business days. The fallback fixing is derived from daily fixings of the fallback overnight index (e.g. SOFR for USD Libor, SONIA for GBP Libor etc.) and a defined fallback spread specific to the index.

Notice that due to the nature of the fallback rate calculation the value is only known for sure at the end of the interest rate period. This is opposed to the old Libor fixing which is known in advance of the interest rate period of the coupon on the fixing date.

Before the fallback rate is fully determined at the end of the interest rate period it comprises both known, historical overnight fixings and projected overnight fixings. Since the projections are computed on the associated index forward curve they will change every day and therefore the estimation of the fallback fixing will change as well from day to day, even if the original Libor index fixing date lies in the past.

8.3 Non-Standard Instrument Pricing

For all non-standard instruments referencing Libor, e.g. swaps referencing Libor in-arrears, caps / floors on Libor, swaptions on Libor, exotics like RPAs on Libor swaps we recommend to change the trade representation so that it references the fallback overnight index directly and includes the fallback spread. This ensures that all features of the trade are accurately taken into account in the pricers.

If the trade representation is not amended, the instrument pricing will still work, but an approximation error will be introduced. As an example consider a cap / floor:

- Without amending the trade representation: The cap payoff is known at the Libor fixing date, before that the projected Libor rate based on the overnight curve and the spread together with the "classic" Black pricer is used to price the cap, the variance is measured from eval date to Libor fixing date.
- With amended trade representation: The cap payoff is known at the end of the Libor interest rate period. The pricing is done as it is implemented for OIS coupons with global cap within the specialized pricer for that.

8.4 Sensitivity Analysis

Trades referencing a Libor rate will show sensitivities either on the Libor tenor or the underlying overnight fallback index depending on the system configuration. The default configuration is the former, i.e. sensitivities stay on the original Libor tenor although the rate estimation is done on the fallback overnight index forward curve.

Trades that are amended, i.e. referencing the fallback overnight index directly will show sensitivities on the overnight index in any case, just as any other "usual" trade referencing overnight indices.

9 Pricing Engine Configuration

The pricing engine configuration file `pricingengine.xml` is provided to select pricing models and pricing engines by product type.

In case of swaptions, an additional field in the envelope of the trade definition can be used to override the pricing product type. Otherwise, the latter is derived from the trade type and economics automatically.

```
<Envelope>
<CounterParty>CPTY_A</CounterParty>
<NettingSetId>CPTY_A</NettingSetId>
<AdditionalFields>
    <pricing_product_type>European_NonStandard</pricing_product_type>
</AdditionalFields>
</Envelope>
```

Listing 243: Explicitly given pricing product type in the envelope.

9.1 Product Type: Ascot

Used by trade type: Ascot

Available Model/Engine pairs:

- BlackScholes/Intrinsic

Engine description:

BlackScholes/Intrinsic builds a IntrinsicAscotEngine. A sample configuration is shown in listing 244.

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="Ascot">
    <Model>BlackScholes</Model>
    <ModelParameters/>
    <Engine>Intrinsic</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">EQ_FD</Parameter>
     </EngineParameters>
</Product>
```

Listing 244: Configuration for Product Ascot, Model: BlackScholes, Engine: Intrinsic

9.2 Product Type: Bond

Used by trade type: Bond

Available Model/Engine pairs:

- DiscountedCashflows/DiscountingRiskyBondEngine
- DiscountedCashflows/DiscountingRiskyBondEngineMultiState

Engine description:

DiscountedCashflows/DiscountingRiskyBondEngine builds a DiscountingRiskyBondEngine. A sample configuration is shown in listing 245.

The parameters have the following meaning:

- OpenEndDateReplacement: if end date is missing in bond schedule (perpetual bond), valuation date plus this period is used as a replacement date
- TimestepPeriod: discretization interval for zero bond pricing
- SensitivityTemplate [optional]: the sensitivity template to use
- IncludePastCashflows [optional]: include past cashflows in the cashflow report, defaults to false (ignored in the DiscountingRiskyBondEngineMultiState)

```
<Product type="Bond">
    <Model>DiscountedCashflows</Model>
    <ModelParameters>
      <Parameter name="OpenEndDateReplacement">50Y</Parameter>
    </ModelParameters>
    <Engine>DiscountingRiskyBondEngine</Engine>
    <EngineParameters>
        <Parameter name="TimestepPeriod">3M</Parameter>
        <Parameter name="SensitivityTemplate">IR_Analytical</Parameter>
        <Parameter name="IncludePastCashflows">true</Parameter>
    </EngineParameters>
</Product>
```

Listing 245: Configuration for Product Bond, Model DiscountedCashflows, Engine DiscountingRiskyBondEngine

DiscountedCashflows/DiscountingRiskyBondEngineMultiState builds a DiscountingRiskyBondEngineMultiState for use in the Credit Model. We refer to the

credit model documentation for further details.

9.3 Product Type: BondOption

Used by trade type: BondOption

Available Model/Engine pairs:

- Black/BlackBondOptionEngine

Engine description:

Black/BlackBondOptionEngine builds a BlackBondOptionEngine. A sample configuration is shown in listing 246.

The parameters have the following meaning:

- TimestepPeriod: discretization interval for zero bond pricing
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="BondOption">
    <Model>Black</Model>
    <ModelParameters/>
    <Engine>BlackBondOptionEngine</Engine>
    <EngineParameters>
        <Parameter name="TimestepPeriod">3M</Parameter>
        <Parameter name="SensitivityTemplate">IR_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 246: Configuration for Product BondOption, Model Black, Engine BlackBondOptionEngine

9.4 Product Type: ConvertibleBond

Used by trade type: ConvertibleBond

Available Model/Engine pairs: DefaultableEquityJumpDiffusion/FD

- DefaultableEquityJumpDiffusion/FD

Engine description:

DefaultableEquityJumpDiffusion/FD builds a FdDefaultableEquityJumpDiffusionConvertibleBondEngine following Andersen, L., and Buffum, D.: Calibration and Implementation of Convertible Bond Models (2002):

The model dynamics for the stock price $S(t)$ is given by

$$dS/S(t^-) = (r(t) - q(t) + \eta h(t, S(t^-)))dt + \sigma(t)dW(t) - \eta dN(t) \tag{213}$$

with a risk free rate $r(t)$, a continuous dividend yield $q(t)$, a default intensity $h(t, S)$, a volatility $\sigma(t)$, a default loss fraction for the equity $\eta \in [0, 1]$ and a Cox process $N(t)$ with

$$E_t(dN(t)) = h(t, S(t^-))dt \tag{214}$$

The notation $S(t^-)$ is shorthand for $\lim_{\epsilon \downarrow 0} S(t-\epsilon)$. The first jump of $N(t)$ represents the default of the equity. See equation (1) in Andersen, Buffum. We support a local default intensity of the form

$$h(t, S(t)) = h_0(t) \left(\frac{S(0)}{S(t)} \right)^p \tag{215}$$

with a deterministic function $h_0(t)$ that is independent from $S(t)$ and a parameter $p \geq 0$. The parameters p and η can be set in the pricing engine configuration. More details on the pricing model is available in a separate model documentation. A sample configuration is shown in listing 247.

The parameters have the following meaning:

- p: the model parameter p
- eta: the model parameter eta
- AdjustEquityForward: If false, the term $\eta h(t, e^z)$ in the coefficient of v_z in the convertible bond pricing pde (see separate model documentation) is set to zero, i.e. the hazard rate h is still used in the discounting term, but the equity drift is not corrected upwards accordingly. The default value is true.
- AdjustEquityVolatility: If false, the market equity volatility input is not adjusted, but directly used in the pricing model. This setting is only possible if $p = 0$. It will then set the weighting with the market survival probability $S(0, t_i)$ in the context of formula for the equity volatility match (see separate model documentation) to zero, i.e. V is taken as the market implied volatility without adjustment. The default value is true.
- AdjustDiscounting: If false, the adjustment of the discounting rate r to the benchmark curve b is suppressed, i.e. the change of the bond pricing PDE to the bond pricing pde with benchmark curve (see separate model documentation) is *not* made (see also section "Curves used in practice" in separate model documentation). The default value is true.
- AdjustCreditSpreadToRR: If true, the credit curve $h(\cdot)$ is adjusted by a factor $\frac{1-R}{1-\rho}$ where R is the recovery rate associated to the market default curve and ρ is the recovery rate of the bond. Usually, $R = \rho$, i.e. the bond recovery rate is the same as the recovery rate of the associated credit curve. In this case, the flag has no effect, since the multiplier is 1. However, when ρ is overwritten with zero due to the flag ZeroRecoveryOverwrite set to true, the flag AdjustCreditSpreadToRR should also be set to true. The default value is false.
- ZeroRecoveryOverwrite: If true, the recovery rate ρ of the convertible bond is overwritten with zero. This option is usually used in conjunction with AdjustCreditSpreadToRR set to true (see below). The default value is false.
- TreatSecuritySpreadAsCreditSpread: If true, the security spread is not incoroporated into the benchmark curve b as described above, but rather added

as a spread on top of the credit curve, i.e. it is added to $h(t, e^z)$. For exchangeables, the security spread is added to *both* h^B and h^S, i.e. it simultaneously increases the credit pread of both the equity and the bond component. Since the security spread is understood as an effective discounting spread, it is scaled by $s \to s/(1-\rho)$ before it is added to h, where ρ is the recovery rate of the bond.

- MesherIsStatic: whether to use the same finite-difference mesher under scenario / sensi calculations
- Bootstrp.CalibrationGrid: The model is calibrated on a configurable set of times . All tenors from the specified grid before the maturity date of the convertible bond are kept and the maturity date itself is added to the resulting grid to avoid calibration for times beyond the bond maturity and at the same time ensuring that we do not need to extrapolate model functions beyond the last calibration point in the pricing.
- Bootstrap.StateGridPoints: The number of state grid points of the Fokker-Planck PDE in the calibration phase
- Bootstrap.MesherEpsilon: The mesher epsilon of the Fokker-Planck PDE in the calibration phase
- Bootstrap.MesherScaling: The mesher scaling multiplier of the Fokker-Planck PDE in the calibration phase
- Bootstrap.Mode: bootstrap strategy: "Simultaneously" or "Alternating", see separate model docs for further details
- Pricing.TimeStepsPerYear: The number of time steps per year to be used for the pricing PDE
- Pricing.StateGridPoints: The number of state grid points to be used for the pricing PDE
- Pricing.MesherEpsilon: The mesher epsilon for the pricing PDE
- Pricing.MesherScaling: The mesher scaling multiplier for the pricing PDE
- ConversionRatioDiscretizationGrid: Multipliers to be used for conversion ratio discretization in the presence of conversion resets / adjustments. See separate model documentation for more details.
- SensitivityTemplate [optional]: the sensitivity template to use

---

```
<Product type="ConvertibleBond">
   <Model>DefaultableEquityJumpDiffusion</Model>
   <ModelParameters>
       <Parameter name="p">0.0</Parameter>
       <Parameter name="eta">1.0</Parameter>
       <Parameter name="AdjustEquityForward">true</Parameter>
       <Parameter name="AdjustEquityVolatility">false</Parameter>
       <Parameter name="AdjustDiscounting">false</Parameter>
       <Parameter name="AdjustCreditSpreadToRR">true</Parameter>
       <Parameter name="ZeroRecoveryOverwrite">true</Parameter>
       <Parameter name="TreatSecuritySpreadAsCreditSpread">true</Parameter>
   </ModelParameters>
```

```
    <Engine>FD</Engine>
    <EngineParameters>
        <Parameter name="MesherIsStatic">true</Parameter>
        <Parameter name="Bootstrap.CalibrationGrid">
          6M,1Y,2Y,3Y,4Y,5Y,7Y,10Y,15Y,20Y,25Y,30Y,40Y,50Y
        </Parameter>
        <Parameter name="Bootstrap.TimeStepsPerYear">24</Parameter>
        <Parameter name="Bootstrap.StateGridPoints">400</Parameter>
        <Parameter name="Bootstrap.MesherEpsilon">1E-5</Parameter>
        <Parameter name="Bootstrap.MesherScaling">1.5</Parameter>
        <Parameter name="Bootstrap.Mode">Alternating</Parameter>
        <Parameter name="Pricing.TimeStepsPerYear">24</Parameter>
        <Parameter name="Pricing.StateGridPoints">100</Parameter>
        <Parameter name="Pricing.MesherEpsilon">1E-4</Parameter>
        <Parameter name="Pricing.MesherScaling">1.5</Parameter>
        <Parameter name="Pricing.ConversionRatioDiscretisationGrid">
          0.5,0.55,0.6,0.65,0.7,0.75,
          0.8,0.85,0.9,0.95,1.0,1.05,
          1.1,1.15,1.2,1.25,1.5,1.75,2.0
        </Parameter>
        <Parameter name="SensitivityTemplate">EQ_FD</Parameter>
    </EngineParameters>
</Product>
```

Listing 247: Configuration for Product ConvertibleBond, Model DefaultableEquityJumpDiffusion, Engine FD

9.5 Product Type: CreditLinkedSwap

Used by trade type: CreditLinkedSwap

Available Model/Engine pairs:

- DiscountedCashflows/DiscountingCreditLinkedSwapEngine

Engine description:

DiscountedCashflows/DiscountingCreditLinkedSwapEngine builds a DiscountingCreditLinkedSwapEngine. A sample configuration is shown in listing 248.

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="CreditLinkedSwap">
    <Model>DiscountedCashflows</Model>
    <ModelParameters/>
    <Engine>DiscountingCreditLinkedSwapEngine</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">IR_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 248: Configuration for Product CreditLinkedSwap, Model DiscountedCashflows, Engine DiscountingCreditLinkedSwapEngine

9.6 Product Type: EuropeanSwaption

Used by trade type: Swaption, for European exercise on vanilla underlying coupon types

Available Model/Engine pairs:

- BlackBachelier/BlackBachelierSwaptionEngine
- LGM/Grid
- LGM/FD
- LGM/MC
- LGM/AMC

Engine description:

BlackBachelier/BlackBachelierSwaptionEngine builds a BlackMultiLegOptionEngine . A sample configuration is shown in listing 249

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="EuropeanSwaption">
    <Model>BlackBachelier</Model>
    <ModelParameters/>
    <Engine>BlackBachelierSwaptionEngine</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">IR_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 249: Configuration for Product EuropeanSwaption, Model BlackBachelier, Engine BlackBachelierSwaptionEngine

LGM/Grid builds a NumericLgmMultiLegOptionEngine using LgmConvoluationSolver as a solver. The rollback follows the paper "Hagan, P: Methodology for callable swaps and Bermudan exercise into swaptions". A sample configuration is shown in listing 250

The parameters have the following meaning:

- Calibration: Bootstrap, BestFit, None
- CalibrationStrategy: CoterminalDealStrike, CoterminalATM, DeltaGammaAdjusted
- ReferenceCalibrationGrid: An optional grid, only one calibration instrument per interval is kept
- Reversion: The mean reversion
- ReversionType: Hagan, HullWhite
- Volatility: The volatility (start value for calibration if calibrated)
- VolatilityType: Hagan, HullWhite
- ShiftHorizon: Shift horizon for LGM model as fraction of deal maturity

- Tolerance: Error tolerance for calibration
- FloatSpreadMapping: mapping of float spreads in analytic swaption pricing for model calibration: proRata, nextCoupon, simple, optional, defaults to proRata.
- sy, sx: Number of covered standard deviations (notation as in Hagan's paper)
- ny, nx: Number of grid points for numerical integration (notation as in Hagan's paper)
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="EuropeanSwaption">
    <Model>LGM</Model>
    <ModelParameters>
        <Parameter name="Calibration">Bootstrap</Parameter>
        <Parameter name="CalibrationStrategy">CoterminalDealStrike</Parameter>
        <Parameter name="ReferenceCalibrationGrid">400,3M</Parameter>
        <Parameter name="Reversion">0.0</Parameter>
        <Parameter name="ReversionType">HullWhite</Parameter>
        <Parameter name="Volatility">0.01</Parameter>
        <Parameter name="VolatilityType">Hagan</Parameter>
        <Parameter name="ShiftHorizon">0.5</Parameter>
        <Parameter name="Tolerance">0.20</Parameter>
        <Parameter name="FloatSpreadMapping">proRata</Parameter>
    </ModelParameters>
    <Engine>Grid</Engine>
    <EngineParameters>
        <Parameter name="sy">5.0</Parameter>
        <Parameter name="ny">30</Parameter>
        <Parameter name="sx">5.0</Parameter>
        <Parameter name="nx">30</Parameter>
        <Parameter name="SensitivityTemplate">IR_FD</Parameter>
    </EngineParameters>
</Product>
```

Listing 250: Configuration for Product EuropeanSwaption, Model LGM, Engine Grid

LGM/FD builds a NumericLgmMultiLegOptionEngine using LgmFdSolver as a solver using finite difference. A sample configuration is shown in listing 251

The parameters have the following meaning:

- Calibration: Bootstrap, BestFit, None
- CalibrationStrategy: CoterminalDealStrike, CoterminalATM, DeltaGammaAdjusted
- ReferenceCalibrationGrid: An optional grid, only one calibration instrument per interval is kept
- Reversion: The mean reversion
- ReversionType: Hagan, HullWhite
- Volatility: The volatility (start value for calibration if calibrated)
- VolatilityType: Hagan, HullWhite
- ShiftHorizon: Shift horizon for LGM model as fraction of deal maturity

- Tolerance: Error tolerance for calibration
- Scheme: The finite difference scheme to use
- StateGridPoints: The number of grid points in state direction
- TimeStepsPerYear: The number of time steps per year to use
- MesherEpsilon: determines the covered probability mass, mass outside state grid is $\Phi^{-1}(1-2\epsilon)$
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="EuropeanSwaption">
    <Model>LGM</Model>
    <ModelParameters>
        <Parameter name="Calibration">Bootstrap</Parameter>
        <Parameter name="CalibrationStrategy">CoterminalDealStrike</Parameter>
        <Parameter name="ReferenceCalibrationGrid">400,3M</Parameter>
        <Parameter name="Reversion">0.0</Parameter>
        <Parameter name="ReversionType">HullWhite</Parameter>
        <Parameter name="Volatility">0.01</Parameter>
        <Parameter name="VolatilityType">Hagan</Parameter>
        <Parameter name="ShiftHorizon">0.5</Parameter>
        <Parameter name="Tolerance">0.20</Parameter>
    </ModelParameters>
    <Engine>FD</Engine>
        <EngineParameters>
            <Parameter name="Scheme">Douglas</Parameter>
            <Parameter name="StateGridPoints">64</Parameter>
            <Parameter name="TimeStepsPerYear">24</Parameter>
            <Parameter name="MesherEpsilon">1E-4</Parameter>
        </EngineParameters>
    </EngineParameters>
</Product>
```

Listing 251: Configuration for Product EuropeanSwaption, Model LGM, Engine FD

LGM/MC builds a McMultiLegOptionEngine. A sample configuration is shown in listing 252

The parameters have the following meaning:

- Calibration: Bootstrap, BestFit, None
- CalibrationStrategy: CoterminalDealStrike, CoterminalATM, DeltaGammaAdjusted
- ReferenceCalibrationGrid: An optional grid, only one calibration instrument per interval is kept
- Reversion: The mean reversion
- ReversionType: Hagan, HullWhite
- Volatility: The volatility (start value for calibration if calibrated)
- VolatilityType: Hagan, HullWhite
- ShiftHorizon: Shift horizon for LGM model as fraction of deal maturity

- Tolerance: Error tolerance for calibration
- Training.Sequence: The sequence type for the traning phase, can be MersenneTwister+, MersenneTwisterAntithetc+, Sobol+, Burley2020Sobol+, SobolBrownianBridge+, Burley2020SobolBrownianBridge+
- Training.Seed: The seed for the random number generation in the training phase
- Training.Samples: The number of samples to be used for the training phase
- Pricing.Sequence: The sequence type for the pricing phase, same values allowed as for training
- Training.BasisFunction: The type of basis function system to be used for the regression analysis, can be Monomial+, Laguerre+, Hermite+, Hyperbolic+, Legendre+, Chbyshev+, Chebyshev2nd+
- BasisFunctionOrder: The order of the basis function system to be used
- Pricing.Seed: The seed for the random number generation in the pricing
- 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)
- BrownianBridgeOrdering: variate ordering for Brownian bridges, can be Steps+, Factors+, Diagonal+
- SobolDirectionIntegers: direction integers for Sobol generator, can be Unit+, Jaeckel+, SobolLevitan+, SobolLevitanLemieux+, JoeKuoD5+, JoeKuoD6+, JoeKuoD7+, Kuo+, Kuo2+, Kuo3+
- 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+
- RegressorModel: Simple, LaggedFX. If not given, it defaults to Simple. Depending on the choice the regressor is built as follows:
  - Simple: For an observation date the full model state observed on this date is included in the regressor. No past states are included though.
  - LaggedFX: For an observation date the full model state observed on this date is included in the regressor. In addition, past FX states that are relevant for future cashflows are included. For example, for a FX resettable cashflow the FX state observed on the FX reset date is included.
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="EuropeanSwaption">
    <Model>LGM</Model>
    <ModelParameters>
        <Parameter name="Calibration">Bootstrap</Parameter>
        <Parameter name="CalibrationStrategy">CoterminalDealStrike</Parameter>
        <Parameter name="ReferenceCalibrationGrid">400,3M</Parameter>
        <Parameter name="Reversion">0.0</Parameter>
        <Parameter name="ReversionType">HullWhite</Parameter>
        <Parameter name="Volatility">0.01</Parameter>
```

```
        <Parameter name="VolatilityType">Hagan</Parameter>
        <Parameter name="ShiftHorizon">0.5</Parameter>
        <Parameter name="Tolerance">0.20</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">0</Parameter>
        <Parameter name="BrownianBridgeOrdering">Steps</Parameter>
        <Parameter name="SobolDirectionIntegers">JoeKuoD7</Parameter>
        <Parameter name="MinObsDate">true</Parameter>
        <Parameter name="RegressorModel">Simple</Parameter>
        <Parameter name="SensitivityTemplate">IR_MC</Parameter>
    </EngineParameters>
</Product>
```

Listing 252: Configuration for Product EuropeanSwaption, Model BlackBachelier, Engine BlackBachelierSwaptionEngine

LGM/AMC builds a McMultiLegOptionEngine for use in AMC simulations. We refer to the AMC module documentation for further details.

9.7 Product Type: EuropeanSwaption NonStandard

A pricing product type similar to European Swaption in subsection 9.6.

The "NonStandard" product type accessed here represents the case of swaptions that do not fullfill the standard case of a fixed-against-floating swaption.

In detail, a swaption is defined to be "Standard" if the following three conditions hold.

- There must be precisely one fixed leg and one floating leg. This condition is necessary since a higher number of legs would make it possible to replicate a varying leg with many constant legs.
- On the fixed leg the values for notional and rate must be constant.
- On the floating leg, the values for notional and spread must be constant. Additional to this, the gearing must be equal to one at all times.

All details from subsection 9.6 are valid here as well.

Generally, in pricingengine.xml both product types (i.e. Standard and NonStandard) can be assigned different pricing models configurations e.g. calibration strategy and pricing models. The user might want to apply the Delta-Gamma-adjusted method for the NonStandard Swaption.

9.8 Product Type: BermudanSwaption

Used by trade type: Swaption, for Bermudan exercise or European exercise on non-vanilla underlying coupon types

Available Model/Engine pairs:

- LGM/Grid
- LGM/FD
- LGM/MC
- LGM/AMC

Engine description:

LGM/Grid builds a NumericLgmMultiLegOptionEngine using LgmConvoluationSolver as a solver. The rollback follows the paper "Hagan, P: Methodology for callable swaps and Bermudan exercise into swaptions". A sample configuration is shown in listing 253

The parameters have the following meaning:

- Calibration: Bootstrap, BestFit, None
- CalibrationStrategy: CoterminalDealStrike, CoterminalATM, DeltaGammaAdjusted
- ReferenceCalibrationGrid: An optional grid, only one calibration instrument per interval is kept
- Reversion: The mean reversion
- ReversionType: Hagan, HullWhite
- Volatility: The volatility (start value for calibration if calibrated)
- VolatilityType: Hagan, HullWhite
- ShiftHorizon: Shift horizon for LGM model as fraction of deal maturity
- Tolerance: Error tolerance for calibration
- sy, sx: Number of covered standard deviations (notation as in Hagan's paper)
- ny, nx: Number of grid points for numerical integration (notation as in Hagan's paper)
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="BermudanSwaption">
    <Model>LGM</Model>
    <ModelParameters>
        <Parameter name="Calibration">Bootstrap</Parameter>
        <Parameter name="CalibrationStrategy">CoterminalDealStrike</Parameter>
        <Parameter name="ReferenceCalibrationGrid">400,3M</Parameter>
        <Parameter name="Reversion">0.0</Parameter>
        <Parameter name="ReversionType">HullWhite</Parameter>
        <Parameter name="Volatility">0.01</Parameter>
        <Parameter name="VolatilityType">Hagan</Parameter>
        <Parameter name="ShiftHorizon">0.5</Parameter>
```

```
        <Parameter name="Tolerance">0.20</Parameter>
    </ModelParameters>
    <Engine>Grid</Engine>
    <EngineParameters>
        <Parameter name="sy">5.0</Parameter>
        <Parameter name="ny">30</Parameter>
        <Parameter name="sx">5.0</Parameter>
        <Parameter name="nx">30</Parameter>
        <Parameter name="SensitivityTemplate">IR_FD</Parameter>
    </EngineParameters>
</Product>
```

Listing 253: Configuration for Product BermudanSwaption, Model LGM, Engine Grid

LGM/FD builds a NumericLgmMultiLegOptionEngine using LgmFdSolver as a solver using finite difference. A sample configuration is shown in listing 254

The parameters have the following meaning:

- Calibration: Bootstrap, BestFit, None
- CalibrationStrategy: CoterminalDealStrike, CoterminalATM, DeltaGammaAdjusted
- ReferenceCalibrationGrid: An optional grid, only one calibration instrument per interval is kept
- Reversion: The mean reversion
- ReversionType: Hagan, HullWhite
- Volatility: The volatility (start value for calibration if calibrated)
- VolatilityType: Hagan, HullWhite
- ShiftHorizon: Shift horizon for LGM model as fraction of deal maturity
- Tolerance: Error tolerance for calibration
- Scheme: The finite difference scheme to use
- StateGridPoints: The number of grid points in state direction
- TimeStepsPerYear: The number of time steps per year to use
- MesherEpsilon: determines the covered probability mass, mass outside state grid is $\Phi^{-1}(1-2\epsilon)$
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="BermudanSwaption">
    <Model>LGM</Model>
    <ModelParameters>
        <Parameter name="Calibration">Bootstrap</Parameter>
        <Parameter name="CalibrationStrategy">CoterminalDealStrike</Parameter>
        <Parameter name="ReferenceCalibrationGrid">400,3M</Parameter>
        <Parameter name="Reversion">0.0</Parameter>
        <Parameter name="ReversionType">HullWhite</Parameter>
        <Parameter name="Volatility">0.01</Parameter>
        <Parameter name="VolatilityType">Hagan</Parameter>
```

```
        <Parameter name="ShiftHorizon">0.5</Parameter>
        <Parameter name="Tolerance">0.20</Parameter>
    </ModelParameters>
    <Engine>FD</Engine>
    <EngineParameters>
        <Parameter name="Scheme">Douglas</Parameter>
        <Parameter name="StateGridPoints">64</Parameter>
        <Parameter name="TimeStepsPerYear">24</Parameter>
        <Parameter name="MesherEpsilon">1E-4</Parameter>
    </EngineParameters>
</Product>
```

Listing 254: Configuration for Product BermudanSwaption, Model LGM, Engine FD

LGM/MC builds a McMultiLegOptionEngine. A sample configuration is shown in listing 255

The parameters have the following meaning:

- Calibration: Bootstrap, BestFit, None
- CalibrationStrategy: CoterminalDealStrike, CoterminalATM, DeltaGammaAdjusted
- ReferenceCalibrationGrid: An optional grid, only one calibration instrument per interval is kept
- Reversion: The mean reversion
- ReversionType: Hagan, HullWhite
- Volatility: The volatility (start value for calibration if calibrated)
- VolatilityType: Hagan, HullWhite
- ShiftHorizon: Shift horizon for LGM model as fraction of deal maturity
- Tolerance: Error tolerance for calibration
- Training.Sequence: The sequence type for the traning phase, can be MersenneTwister+, MersenneTwisterAntithetc+, Sobol+, Burley2020Sobol+, SobolBrownianBridge+, Burley2020SobolBrownianBridge+
- Training.Seed: The seed for the random number generation in the training phase
- Training.Samples: The number of samples to be used for the training phase
- Pricing.Sequence: The sequence type for the pricing phase, same values allowed as for training
- Training.BasisFunction: The type of basis function system to be used for the regression analysis, can be Monomial+, Laguerre+, Hermite+, Hyperbolic+, Legendre+, Chbyshev+, Chebyshev2nd+
- BasisFunctionOrder: The order of the basis function system to be used
- Pricing.Seed: The seed for the random number generation in the pricing

- 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)
- BrownianBridgeOrdering: variate ordering for Brownian bridges, can be Steps+, Factors+, Diagonal+
- SobolDirectionIntegers: direction integers for Sobol generator, can be Unit+, Jaeckel+, SobolLevitan+, SobolLevitanLemieux+, JoeKuoD5+, JoeKuoD6+, JoeKuoD7+, Kuo+, Kuo2+, Kuo3+
- 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+
- RegressorModel: Simple, LaggedFX. If not given, it defaults to Simple. Depending on the choice the regressor is built as follows:
  - Simple: For an observation date the full model state observed on this date is included in the regressor. No past states are included though.
  - LaggedFX: For an observation date the full model state observed on this date is included in the regressor. In addition, past FX states that are relevant for future cashflows are included. For example, for a FX resettable cashflow the FX state observed on the FX reset date is included.
- SensitivityTemplate [optional]: the sensitivity template to use

---

```
<Product type="BermudanSwaption">
    <Model>LGM</Model>
    <ModelParameters>
        <Parameter name="Calibration">Bootstrap</Parameter>
        <Parameter name="CalibrationStrategy">CoterminalDealStrike</Parameter>
        <Parameter name="ReferenceCalibrationGrid">400,3M</Parameter>
        <Parameter name="Reversion">0.0</Parameter>
        <Parameter name="ReversionType">HullWhite</Parameter>
        <Parameter name="Volatility">0.01</Parameter>
        <Parameter name="VolatilityType">Hagan</Parameter>
        <Parameter name="ShiftHorizon">0.5</Parameter>
        <Parameter name="Tolerance">0.20</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">0</Parameter>
        <Parameter name="BrownianBridgeOrdering">Steps</Parameter>
        <Parameter name="SobolDirectionIntegers">JoeKuoD7</Parameter>
        <Parameter name="MinObsDate">true</Parameter>
        <Parameter name="RegressorModel">Simple</Parameter>
        <Parameter name="SensitivityTemplate">IR_MC</Parameter>
    </EngineParameters>
```

```
</Product>
```

Listing 255: Configuration for Product BermudanSwaption, Model BlackBachelier, Engine BlackBachelierSwaptionEngine

LGM/AMC builds a McMultiLegOptionEngine for use in AMC simulations. We refer to the AMC module documentation for further details.

9.9 Product Type: BermudanSwaption NonStandard

A pricing product type very similar to "Bermudan Swaption" in 9.8. The difference between the two types is analogue to the European case, i.e. the difference is as between European Swaption in 9.6 and European Swaption NonStandard in 9.7.

9.10 Product Type: AmericanSwaption

. Used by trade type: Swaption, for American exercise on vanilla underlying coupon types

Available Model/Engine pairs:

- LGM/FD
- LGM/Grid (Not recommended due to inferior performance)
- LGM/MC
- LGM/AMC

Engine description:

LGM/FD builds a NumericLgmMultiLegOptionEngine using LgmFdSolver as a solver using finite difference. A sample configuration is shown in listing 256

The parameters have the following meaning:

- Calibration: Bootstrap, BestFit, None
- CalibrationStrategy: CoterminalDealStrike, CoterminalATM, DeltaGammaAdjusted
- ReferenceCalibrationGrid: An optional grid, only one calibration instrument per interval is kept
- Reversion: The mean reversion
- ReversionType: Hagan, HullWhite
- Volatility: The volatility (start value for calibration if calibrated)
- VolatilityType: Hagan, HullWhite
- ShiftHorizon: Shift horizon for LGM model as fraction of deal maturity
- Tolerance: Error tolerance for calibration
- Scheme: The finite difference scheme to use
- StateGridPoints: The number of grid points in state direction

- TimeStepsPerYear: The number of time steps per year to use
- MesherEpsilon: determines the covered probability mass, mass outside state grid is $\Phi^{-1}(1-2\epsilon)$
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="AmericanSwaption">
    <Model>LGM</Model>
    <ModelParameters>
        <Parameter name="Calibration">Bootstrap</Parameter>
        <Parameter name="CalibrationStrategy">CoterminalDealStrike</Parameter>
        <Parameter name="ReferenceCalibrationGrid">400,3M</Parameter>
        <Parameter name="Reversion">0.0</Parameter>
        <Parameter name="ReversionType">HullWhite</Parameter>
        <Parameter name="Volatility">0.01</Parameter>
        <Parameter name="VolatilityType">Hagan</Parameter>
        <Parameter name="ShiftHorizon">0.5</Parameter>
        <Parameter name="Tolerance">0.20</Parameter>
    </ModelParameters>
    <Engine>FD</Engine>
    <EngineParameters>
        <Parameter name="Scheme">Douglas</Parameter>
        <Parameter name="StateGridPoints">64</Parameter>
        <Parameter name="TimeStepsPerYear">24</Parameter>
        <Parameter name="MesherEpsilon">1E-4</Parameter>
    </EngineParameters>
</Product>
```

Listing 256: Configuration for Product AmericanSwaption, Model LGM, Engine FD

LGM/Grid builds a NumericLgmMultiLegOptionEngine using LgmConvoluationSolver as a solver. The rollback follows the paper "Hagan, P: Methodology for callable swaps and American exercise into swaptions". A sample configuration is shown in listing 257. Not recommended due to inferior performance.

The parameters have the following meaning:

- Calibration: Bootstrap, BestFit, None
- CalibrationStrategy: CoterminalDealStrike, CoterminalATM, DeltaGammaAdjusted
- ReferenceCalibrationGrid: An optional grid, only one calibration instrument per interval is kept
- Reversion: The mean reversion
- ReversionType: Hagan, HullWhite
- Volatility: The volatility (start value for calibration if calibrated)
- VolatilityType: Hagan, HullWhite
- ShiftHorizon: Shift horizon for LGM model as fraction of deal maturity
- Tolerance: Error tolerance for calibration
- sy, sx: Number of covered standard deviations (notation as in Hagan's paper)

- ny, nx: Number of grid points for numerical integration (notation as in Hagan's paper)
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="AmericanSwaption">
    <Model>LGM</Model>
    <ModelParameters>
        <Parameter name="Calibration">Bootstrap</Parameter>
        <Parameter name="CalibrationStrategy">CoterminalDealStrike</Parameter>
        <Parameter name="ReferenceCalibrationGrid">400,3M</Parameter>
        <Parameter name="Reversion">0.0</Parameter>
        <Parameter name="ReversionType">HullWhite</Parameter>
        <Parameter name="Volatility">0.01</Parameter>
        <Parameter name="VolatilityType">Hagan</Parameter>
        <Parameter name="ShiftHorizon">0.5</Parameter>
        <Parameter name="Tolerance">0.20</Parameter>
    </ModelParameters>
    <Engine>Grid</Engine>
    <EngineParameters>
        <Parameter name="sy">5.0</Parameter>
        <Parameter name="ny">30</Parameter>
        <Parameter name="sx">5.0</Parameter>
        <Parameter name="nx">30</Parameter>
        <Parameter name="SensitivityTemplate">IR_FD</Parameter>
    </EngineParameters>
</Product>
```

Listing 257: Configuration for Product AmericanSwaption, Model LGM, Engine Grid (Not recommended due to inferior performance)

LGM/MC builds a McMultiLegOptionEngine. A sample configuration is shown in listing 258

The parameters have the following meaning:

- Calibration: Bootstrap, BestFit, None
- CalibrationStrategy: CoterminalDealStrike, CoterminalATM, DeltaGammaAdjusted
- ReferenceCalibrationGrid: An optional grid, only one calibration instrument per interval is kept
- Reversion: The mean reversion
- ReversionType: Hagan, HullWhite
- Volatility: The volatility (start value for calibration if calibrated)
- VolatilityType: Hagan, HullWhite
- ShiftHorizon: Shift horizon for LGM model as fraction of deal maturity
- Tolerance: Error tolerance for calibration
- Training.Sequence: The sequence type for the traning phase, can be MersenneTwister+, MersenneTwisterAntithetc+, Sobol+, Burley2020Sobol+, SobolBrownianBridge+, Burley2020SobolBrownianBridge+

- Training.Seed: The seed for the random number generation in the training phase
- Training.Samples: The number of samples to be used for the training phase
- Pricing.Sequence: The sequence type for the pricing phase, same values allowed as for training
- Training.BasisFunction: The type of basis function system to be used for the regression analysis, can be Monomial+, Laguerre+, Hermite+, Hyperbolic+, Legendre+, Chbyshev+, Chebyshev2nd+
- BasisFunctionOrder: The order of the basis function system to be used
- Pricing.Seed: The seed for the random number generation in the pricing
- 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)
- BrownianBridgeOrdering: variate ordering for Brownian bridges, can be Steps+, Factors+, Diagonal+
- SobolDirectionIntegers: direction integers for Sobol generator, can be Unit+, Jaeckel+, SobolLevitan+, SobolLevitanLemieux+, JoeKuoD5+, JoeKuoD6+, JoeKuoD7+, Kuo+, Kuo2+, Kuo3+
- 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+
- RegressorModel: Simple, LaggedFX. If not given, it defaults to Simple. Depending on the choice the regressor is built as follows:
  - Simple: For an observation date the full model state observed on this date is included in the regressor. No past states are included though.
  - LaggedFX: For an observation date the full model state observed on this date is included in the regressor. In addition, past FX states that are relevant for future cashflows are included. For example, for a FX resettable cashflow the FX state observed on the FX reset date is included.
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="AmericanSwaption">
    <Model>LGM</Model>
    <ModelParameters>
        <Parameter name="Calibration">Bootstrap</Parameter>
        <Parameter name="CalibrationStrategy">CoterminalDealStrike</Parameter>
        <Parameter name="ReferenceCalibrationGrid">400,3M</Parameter>
        <Parameter name="Reversion">0.0</Parameter>
        <Parameter name="ReversionType">HullWhite</Parameter>
        <Parameter name="Volatility">0.01</Parameter>
        <Parameter name="VolatilityType">Hagan</Parameter>
        <Parameter name="ShiftHorizon">0.5</Parameter>
        <Parameter name="Tolerance">0.20</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">0</Parameter>
        <Parameter name="BrownianBridgeOrdering">Steps</Parameter>
        <Parameter name="SobolDirectionIntegers">JoeKuoD7</Parameter>
        <Parameter name="MinObsDate">true</Parameter>
        <Parameter name="RegressorModel">Simple</Parameter>
        <Parameter name="SensitivityTemplate">IR_MC</Parameter>
    </EngineParameters>
</Product>
```

Listing 258: Configuration for Product AmericanSwaption, Model BlackBachelier, Engine BlackBachelierSwaptionEngine

LGM/AMC builds a McMultiLegOptionEngine for use in AMC simulations. We refer to the AMC module documentation for further details.

9.11 Product Type: AmericanSwaption NonStandard

A pricing product type very similar to "American Swaption" in 9.10. The difference between the two types is analogue to the European case, i.e. the difference is as between European Swaption in 9.6 and European Swaption NonStandard in 9.7.

9.12 Product Type: BondRepo

Used by trade type: BondRepo

Available Model/Engine pairs:

- DiscountedCashflows/DiscountingRepoEngine
- Accrual/AccrualRepoEngine

Engine description:

DiscountedCashflows/DiscountingRepoEngine builds a DiscountingBondRepoEngine. A sample configuration is shown in listing 259.

The parameters have the following meaning:

- IncludeSecurityLeg: include the security leg in the valuation
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="BondRepo">
    <Model>DiscountedCashflows</Model>
    <ModelParameters>
        <Parameter name="IncludeSecurityLeg">true</Parameter>
    </ModelParameters>
    <Engine>DiscountingRepoEngine</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">IR_Analytical</Parameter>
```

```
    </EngineParameters>
</Product>
```

Listing 259: Configuration for Product BondRepo, Model DiscountedCashflows, Engine DiscountingRepoEngine

Accrual/AccrualRepoEngine builds a AccrualBondRepoEngine. A sample configuration is shown in listing 260.

The parameters have the following meaning:

- IncludeSecurityLeg: include the security leg in the valuation
- SensitivityTemplate [optional]: the sensitivity template to use

```
Product type="BondRepo">
    <Model>Accrual</Model>
    <ModelParameters>
        <Parameter name="IncludeSecurityLeg">true</Parameter>
    </ModelParameters>
    <Engine>AccrualRepoEngine</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">IR_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 260: Configuration for Product BondRepo, Model Accrual, Engine DiscountingBondRepoEngine

9.13 Product Type: BondTRS

Used by trade type: BondTRS

Available Model/Engine pairs: DiscountedCashflows/DiscountingBondTRSEngine

Engine description:

DiscountedCashflows/DiscountingBondTRSEngine builds a DisountingBondTRSEngine. A sample configuration is shown in listing 261.

The parameters have the following meaning:

- TreatSecuritySpreadAsCreditSpread [optional]: defaults to false. If true, the security spread is included in the discounting of intermediate bond cashflows.
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="BondTRS">
    <Model>DiscountedCashflows</Model>
    <ModelParameters>
        <Parameter name="TreatSecuritySpreadAsCreditSpread">true</Parameter>
    </ModelParameters>
    <Engine>DiscountingBondTRSEngine</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">IR_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 261: Configuration for Product BondTRS , Model DiscountedCashflows, Engine DiscountingBondTRSEngine

9.14 Product Type: CapFloor

Used by trade type: CapFloor on underlying Ibor with subperiods (all other underlying use their respective coupon pricers)

Available Model/Engine pairs: IborCapModel/IborCapEngine

Engine description:

IborCapModel/IborCapEngine builds a BlackCapFloorEngine or BachelierCapFloorEngine depending on the input volatility type. A sample configuration is shown in listing 262

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="CapFloor">
    <Model>IborCapModel</Model>
    <ModelParameters/>
    <Engine>IborCapEngine</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">IR_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 262: Configuration for Product CapFloor, Model IborCapModel, Engine IborCapEngine

9.15 Product Type: CapFlooredIborLeg

Used by trade type: any trade with a cap / floored ibor / rfr term rate leg

Available Model/Engine pairs: BlackOrBachelier/BlackIborCouponPricer

Engine description:

BlackOrBachelier/BlackIborCouponPricer builds a BlackIborCouponPricer. A sample configuration is shown in listing 263.

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="CapFlooredIborCouponLeg">
    <Model>BlackOrBachelier</Model>
    <ModelParameters/>
    <Engine>BlackIborCouponPricer</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">IR_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 263: Configuration for Product CapFlooredIborLeg, Model BlackOrBachelier, Engine BlackIborCouponPricer

9.16 Product Type: CapFlooredOvernightIndexedCouponLeg

Used by trade type: any trade with a cap / floored OIS leg

Available Model/Engine pairs: BlackOrBachelier/BlackOvernightIndexedCouponPricer

Engine description:

BlackOrBachelier/BlackOvernightIndexedCouponPricer builds a BlackOvernightIndexedCouponPricer. A sample configuration is shown in listing 264.

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="CapFlooredOvernightIndexedCouponLeg">
    <Model>BlackOrBachelier</Model>
    <ModelParameters/>
    <Engine>BlackOvernightIndexedCouponPricer</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">IR_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 264: Configuration for Product CapFlooredOvernightIndexedCouponLeg, Model BlackOrBachelier, Engine BlackOvernightIndexedCouponPricer

9.17 Product Type: CapFlooredAverageONIndexedCouponLeg

Used by trade type: any trade with a cap / floored OIS leg

Available Model/Engine pairs: BlackOrBachelier/BlackAverageONIndexedCouponPricer

Engine description:

BlackOrBachelier/BlackAverageONIndexedCouponPricer builds a BlackAverageONIndexedCouponPricer. A sample configuration is shown in listing 265.

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="CapFlooredAverageONIndexedCouponLeg">
    <Model>BlackOrBachelier</Model>
    <ModelParameters/>
    <Engine>BlackAverageONIndexedCouponPricer</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">IR_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 265: Configuration for Product CapFlooredAverageONIndexedCouponLeg, Model BlackOrBachelier, Engine BlackAverageONIndexedCouponPricer

9.18 Product Type: CapFlooredAverageBMAIndexedCouponLeg

Used by trade type: any trade with a cap / floored OIS leg

Available Model/Engine pairs:
BlackOrBachelier/BlackAverageBMAIndexedCouponPricer

Engine description:

BlackOrBachelier/BlackAverageBMAIndexedCouponPricer builds a BlackAverageBMAIndexedCouponPricer. A sample configuration is shown in listing 266.

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="CapFlooredAverageBMAIndexedCouponLeg">
    <Model>BlackOrBachelier</Model>
    <ModelParameters/>
    <Engine>BlackAverageBMAIndexedCouponPricer</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">IR_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 266: Configuration for Product CapFlooredAverageBMAIndexedCouponLeg, Model BlackOrBachelier, Engine BlackAverageBMAIndexedCouponPricer

9.19 Product Type: CappedFlooredCpiLegCoupons

Used by trade type: any trade with a cap / floored CPI leg (coupons)

Available Model/Engine pairs: Black/BlackAnalytic

Engine description:

Black/BlackAnalytic builds a BlackCPICouponPricer or BachelierCPICouponPricer, depending on the volatility input. A sample configuration is shown in listing 267

The parameters have the following meaning:

- useLastFixingDate: if true, use the last known fixing date as the base date of the volatility structure, otherwise use observation lag
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="CapFlooredCpiLegCoupons">
    <Model>Black</Model>
    <ModelParameters/>
    <Engine>BlackAnalytic</Engine>
```

```
    <EngineParameters>
          <Parameter name="useLastFixingDate">true</Parameter>
      <Parameter name="SensitivityTemplate">IR_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 267: Configuration for Product CapFlooredCpiLegCoupons, Model Black, Engine Black-Analytic

9.20 Product Type: CappedFlooredCpiLegCashFlows

Used by trade type: any trade with a cap / floored CPI leg (cashflows)

Available Model/Engine pairs: Black/BlackAnalytic

Engine description:

Black/BlackAnalytic builds a BlackCPICashFLowPricer or BachelierCPICashFlowPricer, depending on the volatility input. A sample configuration is shown in listing 268

The parameters have the following meaning:

- useLastFixingDate: if true, use the last known fixing date as the base date of the volatility structure, otherwise use observation lag
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="CappedFlooredCpiLegCashFlows">
    <Model>Black</Model>
    <ModelParameters/>
    <Engine>BlackAnalytic</Engine>
    <EngineParameters>
          <Parameter name="useLastFixingDate">true</Parameter>
      <Parameter name="SensitivityTemplate">IR_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 268: Configuration for Product CapFlooredCpiLegCoupons, Model Black, Engine Black-Analytic

9.21 Product Type: CommodityAveragePriceOption

Used by trade type: CommodityAveragePriceOption

Available Model/Engine pairs:

- Black/AnalyticalApproximation
- Black/MonteCarlo

Engine description:

Black/AnalyticalApproximation builds a CommodityAveragePriceOptionAnalyticalEngine. The correlation between two future contracts is parametrized as

$$\rho(s,t) = e^{-\beta|s-t|}$$

where s and t are times to futures expiry. A sample configuration is shown in listing 269.

The parameters have the following meaning:

- beta: parameter in correlation parametrization
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="CommodityAveragePriceOption">
    <Model>Black</Model>
    <ModelParameters/>
    <Engine>AnalyticalApproximation</Engine>
    <EngineParameters>
        <Parameter name="beta">0</Parameter>
        <Parameter name="SensitivityTemplate">COMM_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 269: Configuration for Product CommodityAveragePriceOption, Model Black, Engine AnalyticalApproximation

Black/MonteCarlo builds a CommodityAveragePriceOptionMonteCarloEngine. A sample configuration is shown in listing 270.

The parameters have the following meaning:

- samples: the number of Monte Carlo Samples
- beta: parameter in correlation parametrization
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="CommodityAveragePriceOption">
    <Model>Black</Model>
    <ModelParameters/>
    <Engine>MonteCarlo</Engine>
    <EngineParameters>
        <Parameter name="samples">10000</Parameter>
        <Parameter name="beta">0</Parameter>
        <Parameter name="SensitivityTemplate">COMM_MC</Parameter>
    </EngineParameters>
</Product>
```

Listing 270: Configuration for Product CommodityAveragePriceOption, Model Black, Engine MonteCarlo

9.22 Product Type: CommodityAveragePriceBarrierOption

Used by trade type: CommodityAveragePriceOption with BarrierData

Available Model/Engine pairs:

- Black/MonteCarlo

Engine description:

Black/MonteCarlo builds a CommodityAveragePriceOptionAnalyticalEngine. The correlation between two future contracts is parametrized as

$$\rho(s,t) = e^{-\beta|s-t|}$$

where s and t are times to futures expiry. A sample configuration is shown in listing 271.

The parameters have the following meaning:

- samples: the number of Monte Carlo Samples
- beta: parameter in correlation parametrization
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="CommodityAveragePriceBarrierOption">
    <Model>Black</Model>
    <ModelParameters/>
    <Engine>MonteCarlo</Engine>
    <EngineParameters>
        <Parameter name="samples">10000</Parameter>
        <Parameter name="beta">0</Parameter>
        <Parameter name="SensitivityTemplate">COMM_MC</Parameter>
    </EngineParameters>
</Product>
```

Listing 271: Configuration for Product CommodityAveragePriceBarrierOption, Model Black, Engine MonteCarlo

9.23 Product Type: CommodityForward

Used by trade type: CommodityForward

Available Model/Engine pairs:

- DiscountedCashflows/DiscountingCommodityForwardEngine

Engine description:

DiscountedCashflows/DiscountingCommodityForwardEngine builds a DiscountingCommodityForwardEngine. A sample configuration is shown in listing 272.

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="CommodityForward">
    <Model>DiscountedCashflows</Model>
    <ModelParameters/>
    <Engine>DiscountingCommodityForwardEngine</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">COMM_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 272: Configuration for Product CommodityForward, Model DiscountedCashflows, Engine DiscountingCommodityForwardEngine

9.24 Product Type: CreditDefaultSwap

Used by trade type: CreditDefaultSwap

Available Model/Engine pairs:

- DiscountedCashflows/MidPointCdsEngine
- DiscountedCashflows/MidPointCdsEngineMultiState

Engine description:

DiscountedCashflows/MidPointCdsEngine builds a MidPointCdsEngine. A sample configuration is shown in listing 273.

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="CreditDefaultSwap">
    <Model>DiscountedCashflows</Model>
    <ModelParameters/>
    <Engine>MidPointCdsEngine</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">IR_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 273: Configuration for Product CreditDefaultSwap, Model DiscountedCashflows, Engine MidPointCdsEngine

DiscountedCashflows/MidPointCdsEngineMultiState builds a MidPointCdsEngineMultiState. This engine is only used in the context of the Credit Model. We refer to the documentation of this module for further details.

9.25 Product Type: CreditDefaultSwapOption

Used by trade type: CreditDefaultSwapOption

Available Model/Engine pairs:

- Black/BlackCdsOptionEngine

Engine description:

Black/BlackCdsOptionEngine builds a BlackCdsOptionEngine. A sample configuration is shown in listing 274.

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="CreditDefaultSwapOption">
    <Model>Black</Model>
```

```
    <ModelParameters/>
    <Engine>BlackCdsOptionEngine</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">IR_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 274: Configuration for Product CreditDefaultSwap, Model DiscountedCashflows, Engine MidPointCdsEngine

9.26 Product Type: IndexCreditDefaultSwap

Used by trade type: IndexCreditDefaultSwap

Available Model/Engine pairs:

- DiscountedCashflows/MidPointIndexCdsEngine

Engine description:

DiscountedCashflows/MidPointIndexCdsEngine builds a MidPointIndexCdsEngine. A sample configuration is shown in listing 275.

The parameters have the following meaning:

- Curve: Index, Underlying
- SensitivityDecomposition: Underlying, NotionalWeighted, LossWeighted, DeltaWeighted
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="IndexCreditDefaultSwap">
    <Model>DiscountedCashflows</Model>
    <ModelParameters/>
    <Engine>MidPointIndexCdsEngine</Engine>
    <EngineParameters>
        <Parameter name="Curve">Index</Parameter>
        <Parameter name="SensitivityDecomposition">DeltaWeighted</Parameter>
        <Parameter name="SensitivityTemplate">IR_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 275: Configuration for Product CreditDefaultSwap, Model DiscountedCashflows, Engine MidPointIndexCdsEngine

9.27 Product Type: IndexCreditDefaultSwapOption

Used by trade type: IndexCreditDefaultSwapOption

Available Model/Engine pairs:

- Black/BlackIndexCdsOptionEngine
- LognormalAdjustedIndexSpread/NumericalIntegrationEngine

Engine description:

Black/BlackIndexCdsOptionEngine builds a BlackIndexCdsOptionEngine. A sample configuration is shown in listing 276.

The parameters have the following meaning:

- Curve: Index, Underlying
- FepCurve: Index, Underlying
- SensitivityDecomposition: Underlying, NotionalWeighted, LossWeighted, DeltaWeighted
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="IndexCreditDefaultSwapOption">
    <Model>DiscountedCashflows</Model>
    <ModelParameters/>
    <Engine>MidPointIndexCdsEngine</Engine>
    <EngineParameters>
        <Parameter name="Curve">Index</Parameter>
        <Parameter name="FepCurve">Index</Parameter>
        <Parameter name="SensitivityDecomposition">DeltaWeighted</Parameter>
        <Parameter name="SensitivityTemplate">IR_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 276: Configuration for Product CreditDefaultSwap, Model DiscountedCashflows, Engine MidPointIndexCdsEngine

LognormalAdjustedIndexSpread/NumericalIntegrationEngine builds a NumericalIntegrationIndexCdsOptionEngine. A sample configuration is shown in listing 277.

The parameters have the following meaning:

- Curve: Index, Underlying
- FepCurve: Index, Underlying
- SensitivityDecomposition: Underlying, NotionalWeighted, LossWeighted, DeltaWeighted
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="IndexCreditDefaultSwapOption">
    <Model>DiscountedCashflows</Model>
    <ModelParameters/>
    <Engine>MidPointIndexCdsEngine</Engine>
    <EngineParameters>
        <Parameter name="Curve">Index</Parameter>
        <Parameter name="FepCurve">Index</Parameter>
        <Parameter name="SensitivityDecomposition">DeltaWeighted</Parameter>
        <Parameter name="SensitivityTemplate">IR_Semianalytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 277: Configuration for Product CreditDefaultSwap, Model DiscountedCashflows, Engine MidPointIndexCdsEngine

9.28 Product Type: CpiCapFloor

Used by trade type: CapFloor with underlying leg of leg type CPI.

Available Model/Engine pairs: CpiCapModel/CpiCapEngine

Engine description:

CpiCapModel/CpiCapEngine builds a CPIBlackCapFloorEngine or CPIBachelierCapFloorEngine depneding on the volatility type of the market surface. A sample configuration is shown in listing 278.

The parameters have the following meaning:

- useLastFixingDate: if true, use the last known fixing date as the base date of the volatility structure, otherwise use observation lag
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="CpiCapFloor">
    <Model>CpiCapModel</Model>
    <ModelParameters/>
    <Engine>CpiCapEngine</Engine>
    <EngineParameters>
        <Parameter name="useLastFixingDate">true</Parameter>
        <Parameter name="SensitivityTemplate">IR_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 278: Configuration for Product CpiCapFloor, Model CpiCapModel, Engine CpiCapEngine

9.29 Product Type: YYCapFloor

Used by trade type: CapFloor with underlying leg of leg type YoY.

Available Model/Engine pairs: YYCapModel/YYCapEngine

Engine description:

YYCapModel/YYCapEngine builds a YoYInflationBlackCapFloorEngine, YoYInflationUnitDisplacedBlackCapFloorEngine, YoYInflationBachelierCapFloorEngine depending on the volatility type of the market surface. A sample configuration is shown in listing 279.

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="CpiCapFloor">
    <Model>YYCapModel</Model>
    <ModelParameters/>
    <Engine>YYCapEngine</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">IR_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 279: Configuration for Product YYCapFloor, Model YYCapModel, Engine YYCapEngine

9.30 Product Type: CappedFlooredYYLeg

Used by trade type: any trade with a cap / floored YY leg

Available Model/Engine pairs: CapFlooredYYModel/CapFlooredYYCouponPricer

Engine description:

CapFlooredYYModel/CapFlooredYYCouponPricer builds a BlackYoYInflationCouponPricer, UnitDisplacedBlackYoYInflationCouponPricer, BachelierYoYInflationCouponPricer, depending on the volatility input. A sample configuration is shown in listing 280

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="CappedFlooredYYLeg">
    <Model>CapFlooredYYModel</Model>
    <ModelParameters/>
    <Engine>CapFlooredYYCouponPricer</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">IR_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 280: Configuration for Product CapFlooredYYLeg, Model CapFlooredYYModel, Engine CapFlooredYYCouponPricer

9.31 Product Type: CappedFlooredNonStdYYLeg

Used by trade type: any trade with a cap / floored YY leg if IrregularYoY is true in YoY leg data

Available Model/Engine pairs: CapFlooredYNonStdYYModel/CapFlooredNonStdYYCouponPricer

Engine description:

CapFlooredYNonStdYYModel/CapFlooredNonStdYYCouponPricer builds a NonStandardBlackYoYInflationCouponPricer, NonStandardUnitDisplacedBlackYoYInflationCouponPricer, NonStandardBachelierYoYInflationCouponPricer, depending on the volatility input. A sample configuration is shown in listing 281

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="CappedFlooredNonStdYYLeg">
    <Model>CapFlooredNonStdYYModel</Model>
    <ModelParameters/>
```

```
    <Engine>CapFlooredNonStdYYCouponPricer</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">IR_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 281: Configuration for Product CapFlooredNonStdYYLeg, Model CapFlooredNonStdYYModel, Engine CapFlooredNonStdYYCouponPricer

9.32 Product Type: CMS

Used by trade type: any trade referencing a CMS leg, also used by CMS Spread coupon pricers

Available Model/Engine pairs:

- LinearTSR/LinearTSRPricer
- Hagan/Analytic
- Hagan/Numerical

Engine description:

LinearTSR/LinearTSRPricer builds a LinearTRSPricer. A sample configuration is shown in listing 282.

The parameters have the following meaning:

- MeanReversion: the mean reversion for the model
- Policy: RateBound, VegaRatio, PriceThreshold, BsStdDev
- LowerRateBoundLogNormal, UpperRateBoundLogNormal: rate bounds for ln / sln vol input
- LowerRateNormal, UpperRateNormal: rate bounds for normal vol input
- VegaRatio: vega ratio for policy
- PriceThreshold: price threshold for policy
- BsStdDev: std devs for BsStdDev
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Prooduct type="CMS">
    <Model>LinearTSR</Model>
    <ModelParameters/>
    <Engine>LinearTSRPricer</Engine>
    <EngineParameters>
        <Parameter name="MeanReversion">0.0</Parameter>
        <Parameter name="Policy">RateBound</Parameter>
        <Parameter name="LowerRateBoundLogNormal">0.0001</Parameter>
        <Parameter name="UpperRateBoundLogNormal">2.0000</Parameter>
        <Parameter name="LowerRateBoundNormal">-2.0000</Parameter>
        <Parameter name="UpperRateBoundNormal">2.0000</Parameter>
        <Parameter name="VegaRatio">0.01</Parameter>
        <Parameter name="PriceThreshold">0.0000001</Parameter>
        <Parameter name="BsStdDev">3.0</Parameter>
```

```
        <Parameter name="SensitivityTemplate">IR_Semianalytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 282: Configuration for Product CMS, Model LinearTSR, Engine LinearTSRPricer

Hagan/Analytic, Hagan/Numerical build AnalyticHaganPricer, NumericHaganPricer (TODO add parameters and sample config).

9.33 Product Type: SyntheticCDO

Used by trade type: SyntheticCDO

Available Model/Engine pairs: GaussCopula/Bucketing and GaussCopula/MonteCarlo

Engine description:

GaussCopula/Bucketing builds a IndexCdsTrancheEngine. A sample configuration is shown in listing 283 It uses a Bucketing Algorithm to compute the expected tranche loss

GaussCopula/MonteCarlo builds a IndexCdsTrancheEngine. The expected tranche loss is computed using a Monte Carlo simulation.

The parameters have the following meaning:

- min, max: min max std dev for Gauss copula
- steps: integration steps
- useStochasticRecovery: whether to use deterministic (false) or stochastic (true) recovery model
- useAssumedRecovery: wheter to use specialized default curves, which are bootstrapped using the assumed recovery rate
- recoveryRateGrid: recovery rate grid for the recovery rate grid
- reocveryRateProbabilities: recovery rate probabilities for the recovery rate grid
- buckets: number of buckets in Hull-White bucketing
- nSimulations: number of simulations in the Monte Carlo Simulation
- SensitivityDecomposition: Underlying, NotionalWeighted, LossWeighted, DeltaWeighted
- useLossDistWhenJustified: whether to use QuantLib::LossDist for determinisitc recovery instead of HulWhiteBucketing
- homogeneousPoolWhenJustified: whether to use homogeneous pool if possible, applies to QuantLib::LossDist
- useQuadrature: whether to use quadrature
- calibrateConstituentCurves: whether to calibrate constituent curves to index level
- SensitivityTemplate [optional]: the sensitivity template to use

- useIndexCDSPricer [optional]: whether to use the index CDS pricer for tranches with detachment point at 100 percent, defaults to true
- useIndexCurveForIndexCDS [optional]: use the index curve for the 100 tranche instead of the constituent curve, defaults to false

It is possible to define multiple recovery rate grids and probabilities. For example, you can define a grid for different CDS tiers, such as SNRFOR. The parameter name for a specific grid is recoveryRateGrid_SNRFOR, and the corresponding probabilities are recoveryRateProbabilities_SNRFOR.

If you need to differentiate the grids further based on the underlying, such as having different grids for CDX HY and CDX IG, you can add the *IndexSubFamily* identifier to the index credit reference data. This allows you to set up independent grids for groups of credit indices, such as a grid for CDX High Yields Tranches recoveryRateGrid_CDX.NA.HY_SNRFOR.

The recovery rate grid and probabilities are defined as a comma-separated list of values. The previous allowed values *Markit2020* and *constant* are deprecated and will cause a warning. With the setting *Markit2020* there will be a recovery grid with three points used. The grid will be be based around the constituent's recovery. If the recovery rate is greater than 10 % and lower than 55 %, the grid is $[0.1, RR, 2RR - 0.1]$, otherwise a constant recovery rate will be assumed.

```
<Product type="SyntheticCDO">
   <Model>GaussCopula</Model>
   <ModelParameters>
       <Parameter name="min">-5.0</Parameter>
       <Parameter name="max">5.0</Parameter>
       <Parameter name="steps">64</Parameter>
       <Parameter name="useStochasticRecovery">Y</Parameter>
       <Parameter name="useAssumedRecovery">Y</Parameter>
       <Parameter name="recoveryRateGrid">0.7,0.4,0.1</Parameter>
       <Parameter name="recoveryRateProbabilities">0.35,0.3,0.35</Parameter>
       <Parameter name="recoveryRateGrid_SUBLT2">0.2</Parameter>
       <Parameter name="recoveryRateProbabilities_SUBLT2">1.0</Parameter>
       <Parameter name="recoveryRateGrid_SNRLAC">0.4</Parameter>
       <Parameter name="recoveryRateProbabilities_SNRLAC">1.0</Parameter>
       <Parameter name="recoveryRateGrid_SECDOM">0.4</Parameter>
       <Parameter name="recoveryRateProbabilities_SECDOM">1.0</Parameter>
       <Parameter name="recoveryRateGrid_CDX.NA.HY_SECDOM">0.5,0.3,0.1</Parameter>
       <Parameter name="recoveryRateProbabilities_CDX.NA.HY_SECDOM">0.35,0.3,0.35</Parameter>
       <Parameter name="recoveryRateGrid_CDX.NA.HY_SNRFOR">0.5,0.3,0.1</Parameter>
       <Parameter name="recoveryRateProbabilities_CDX.NA.HY_SNRFOR">0.35,0.3,0.35</Parameter>
   </ModelParameters>
   <Engine>Bucketing</Engine>
   <EngineParameters>
       <Parameter name="buckets">124</Parameter>
       <Parameter name="SensitivityDecomposition">DeltaWeighted</Parameter>
       <Parameter name="useLossDistWhenJustified">N</Parameter>
       <Parameter name="homogeneousPoolWhenJustified">N</Parameter>
       <Parameter name="calibrateConstituentCurves">N</Parameter>
       <Parameter name="calibrationIndexTerms">3Y,5Y</Parameter>
       <Parameter name="SensitivityTemplate">CR_Semianalytical</Parameter>
       <Parameter name="useIndexCDSPricer">Y</Parameter>
       <Parameter name="useIndexCurveForIndexCDS">N</Parameter>
```

```
    </EngineParameters>
</Product>
```

Listing 283: Configuration for Product SyntheticCDO, Model GaussCopula, Engine Bucketing

9.34 Product Type: CBO

Used by trade type: CBO

Available Model/Engine pairs: OneFactorCopula/MonteCarloCBOEngine

Engine description:

OneFactorCopula/MonteCarloCBOEngine builds a MonteCarloCBOEngine. A sample configuration is shown in listing 284

The parameters have the following meaning:

- Samples: number of MC samples
- Bins: number of bins used for discretization
- Seed: seed for MC simulation
- LossDistributionPeriods:
- Correlation: correlation to use
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="CBO">
    <Model>OneFactorCopula</Model>
    <ModelParameters/>
    <Engine>MonteCarloCBOEngine</Engine>
    <EngineParameters>
        <Parameter name="Samples">1000</Parameter>
        <Parameter name="Bins">20</Parameter>
        <Parameter name="Seed">42</Parameter>
        <Parameter name="LossDistributionPeriods"/>
        <Parameter name="Correlation">0.2</Parameter>
        <Parameter name="SensitivityTemplate">CR_MC</Parameter>
    </EngineParameters>
</Product>
```

Listing 284: Configuration for Product CBO, Model OneFactorCopula, Engine MonteCarloCBOEngine

9.35 Product Type: CMSSpread

Used by trade type: any trade referencing a CMSSpread leg

Available Model/Engine pairs:

- BrigoMercurio/Analytic

Engine description:

BrigoMercurio/Analytic builds a LognormalCmsSpreadPricer (following Brigo, Mercurio, Interest Rate Models - Theory and Practice, section 13.16.2). A sample configuration is shown in listing 285.

The parameters have the following meaning:

- IntegrationPoints: Number of points for Gauss-Hermite numerical integration
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="CMSSpread">
    <Model>BrigoMercurio</Model>
    <ModelParameters/>
    <Engine>Analytic</Engine>
    <EngineParameters>
        <Parameter name="IntegrationPoints">16</Parameter>
        <Parameter name="SensitivityTemplate">IR_Semianalytical</Parameter>
    </EngineParameters>
</Product>\caption{Configuration for Product CMSSpread, Model BrigoMercurio, Engine Analytic}
```

Listing 285: Configuration for Product CMSSpread, Model BrigoMercurio, Engine Analytic

9.36 Product Type: DurationAdjustedCMS

Used by trade type: any trade referencing a DurationAdjustedCMS leg

Available Model/Engine pairs:

- LinearTSR/LinearTSRPricer

Engine description:

LinearTSR/LinearTSRPricer builds a DurationAdjustedCmsCouponTsrPricer with LinearAnnuityMapping. A sample configuration is shown in listing 286.

The parameters have the following meaning:

- MeanReversion: the mean reversion for the model
- LowerRateBoundLogNormal, UpperRateBoundLogNormal: rate bounds for ln / sln vol input
- LowerRateNormal, UpperRateNormal: rate bounds for normal vol input
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="DurationAdjustedCMS">
    <Model>LinearTSR</Model>
    <ModelParameters/>
    <Engine>LinearTSRPricer</Engine>
    <EngineParameters>
        <Parameter name="MeanReversion">0.0</Parameter>
        <Parameter name="LowerRateBoundLogNormal">0.0001</Parameter>
        <Parameter name="UpperRateBoundLogNormal">2.0000</Parameter>
        <Parameter name="LowerRateBoundNormal">-2.0000</Parameter>
        <Parameter name="UpperRateBoundNormal">2.0000</Parameter>
        <Parameter name="SensitivityTemplate">IR_Semianalytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 286: Configuration for Product DurationAdjustedCMS, Model LinearTSR, Engine LinearTSRPricer

9.37 Product Type: CommodityAsianOptionArithmeticPrice

Used by trade type: CommodityAsianOption if payoffType2 is Arithmetric and payoffType is Asian

Available Model/Engine pairs:

- BlackScholesMerton/MCDiscreteArithmeticAPEngine
- BlackScholesMerton/TurnbullWakemanAsianEngine
- ScriptedTrade/ScriptedTrade

Engine description:

BlackScholesMerton/MCDiscreteArithmeticAPEngine builds a MCDiscreteArithmeticAPEngine using Sobol sequences. A sample configuration is shown in listing 287.

The parameters have the following meaning:

- BrownianBridge: whether to use Brownian Bridge for MC simulation
- AntitheticVariate: whether to use antithetic variates for MC simulation
- ControlVariate: whether to use control variate for MC simulation
- RequiredSamples: minimum number of samples for MC simulation, if 0 (not specified) RequiredTolerance must be given
- RequiredTolernace: max tolerance for MC error, if 0 (not specified) RequiredSamples must be given
- MaxSamples: max number of samples for MC simulation
- Seed: seed for random number generator
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="CommodityAsianOptionArithmeticPrice">
    <Model>BlackScholesMerton</Model>
    <ModelParameters/>
    <Engine>MCDiscreteArithmeticAPEngine</Engine>
    <EngineParameters>
        <Parameter name="BrownianBridge">true</Parameter>
        <Parameter name="AntitheticVariate">true</Parameter>
        <Parameter name="ControlVariate">true</Parameter>
        <Parameter name="RequiredSamples">10000</Parameter>
        <Parameter name="RequiredTolerance">0</Parameter>
        <Parameter name="MaxSamples">0</Parameter>
        <Parameter name="Seed">42</Parameter>
        <Parameter name="SensitivityTemplate">COMM_MC</Parameter>
    </EngineParameters>
</Product>
```

Listing 287: Configuration for Product CommodityAsianOptionArithmeticPrice, Model BlackScholesMerton, Engine MCDiscreteArithmeticAPEngine

BlackScholesMerton/TurnbullWakemanAsianEngine builds a TurnbullWakemanAsianEngine. A sample configuration is shown in listing 288.

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="CommodityAsianOptionArithmeticPrice">
    <Model>BlackScholesMerton</Model>
    <ModelParameters/>
    <Engine>TurnbullWakemanAsianEngine</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">COMM_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 288: Configuration for Product CommodityAsianOptionArithmeticPrice, Model BlackScholesMerton, Engine TurnbullWakemanAsianEngine

ScriptedTrade/ScriptedTrade delegates to the scripted trade engine and the associated pricing engine configuration for Product Type ScriptedTrade, see there for details. A sample configuration is given in listing 289.

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="CommodityAsianOptionArithmeticPrice">
    <Model>ScriptedTrade</Model>
    <ModelParameters/>
    <Engine>ScriptedTrade</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">COMM_MC</Parameter>
    </EngineParameters>
</Product>
```

Listing 289: Configuration for Product CommodityAsianOptionArithmeticPrice, Model ScriptedTrade, Engine ScriptedTrade

9.38 Product Type: CommodityAsianOptionArithmeticStrike

Used by trade type: CommodityAsianOption if payoffType2 is Arithmetric and payoffType is AverageStrike

Available Model/Engine pairs:

- BlackScholesMerton/MCDiscreteArithmeticASEngine
- ScriptedTrade/ScriptedTrade

Engine description:

BlackScholesMerton/MCDiscreteArithmeticASEngine builds a

MCDiscreteArithmeticASEngine using Sobol sequences. A sample configuration is shown in listing 290.

The parameters have the following meaning:

- BrownianBridge: whether to use Brownian Bridge for MC simulation
- AntitheticVariate: whether to use antithetic variates for MC simulation
- ControlVariate: whether to use control variate for MC simulation
- RequiredSamples: minimum number of samples for MC simulation, if 0 (not specified) RequiredTolerance must be given
- RequiredTolernace: max tolerance for MC error, if 0 (not specified) RequiredSamples must be given
- MaxSamples: max number of samples for MC simulation
- Seed: seed for random number generator
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="CommodityAsianOptionArithmeticStrike">
    <Model>BlackScholesMerton</Model>
    <ModelParameters/>
    <Engine>MCDiscreteArithmeticASEngine</Engine>
    <EngineParameters>
        <Parameter name="BrownianBridge">true</Parameter>
        <Parameter name="AntitheticVariate">true</Parameter>
        <Parameter name="ControlVariate">true</Parameter>
        <Parameter name="RequiredSamples">10000</Parameter>
        <Parameter name="RequiredTolerance">0</Parameter>
        <Parameter name="MaxSamples">0</Parameter>
        <Parameter name="Seed">42</Parameter>
        <Parameter name="SensitivityTemplate">COMM_MC</Parameter>
    </EngineParameters>
</Product>
```

Listing 290: Configuration for Product CommodityAsianOptionArithmeticStrike, Model BlackScholesMerton, Engine MCDiscreteArithmeticASEngine

ScriptedTrade/ScriptedTrade delegates to the scripted trade engine and the associated pricing engine configuration for Product Type ScriptedTrade, see there for details. A sample configuration is given in listing 291.

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="CommodityAsianOptionArithmeticStrike">
    <Model>ScriptedTrade</Model>
    <ModelParameters/>
    <Engine>ScriptedTrade</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">COMM_MC</Parameter>
    </EngineParameters>
</Product>
```

Listing 291: Configuration for Product CommodityAsianOptionArithmeticStrike, Model ScriptedTrade, Engine ScriptedTrade

9.39 Product Type: CommodityAsianOptionGeometricPrice

Used by trade type: CommodityAsianOption if payoffType2 is Geometric and payoffType is Asian

Available Model/Engine pairs:

- BlackScholesMerton/MCDiscreteGeometricAPEngine
- BlackScholesMerton/AnalyticDiscreteGeometricAPEngine
- BlackScholesMerton/AnalyticContinuousGeometricAPEngine
- ScriptedTrade/ScriptedTrade

Engine description:

BlackScholesMerton/MCDiscreteGeometricAPEngine builds a MCDiscreteGeometricAPEngine using Sobol sequences. A sample configuration is shown in listing 292.

The parameters have the following meaning:

- BrownianBridge: whether to use Brownian Bridge for MC simulation
- AntitheticVariate: whether to use antithetic variates for MC simulation
- ControlVariate: whether to use control variate for MC simulation
- RequiredSamples: minimum number of samples for MC simulation, if 0 (not specified) RequiredTolerance must be given
- RequiredTolernace: max tolerance for MC error, if 0 (not specified) RequiredSamples must be given
- MaxSamples: max number of samples for MC simulation
- Seed: seed for random number generator
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="CommodityAsianOptionGeometricPrice">
    <Model>BlackScholesMerton</Model>
    <ModelParameters/>
    <Engine>MCDiscreteGeometricAPEngine</Engine>
    <EngineParameters>
        <Parameter name="BrownianBridge">true</Parameter>
        <Parameter name="AntitheticVariate">true</Parameter>
        <Parameter name="ControlVariate">true</Parameter>
        <Parameter name="RequiredSamples">10000</Parameter>
        <Parameter name="RequiredTolerance">0</Parameter>
        <Parameter name="MaxSamples">0</Parameter>
        <Parameter name="Seed">42</Parameter>
        <Parameter name="SensitivityTemplate">COMM_MC</Parameter>
```

```
    </EngineParameters>
</Product>
```

Listing 292: Configuration for Product CommodityAsianOptionGeometricPrice, Model BlackScholesMerton, Engine MCDiscreteGeometricAPEngine

BlackScholesMerton/AnalyticDiscreteGeometricAPEngine builds a AnalyticDiscreteGeometricAveragePriceAsianEngine. A sample configuration is shown in listing 293.

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="CommodityAsianOptionGeometricPrice">
    <Model>BlackScholesMerton</Model>
    <ModelParameters/>
    <Engine>AnalyticDiscreteGeometricAPEngine</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">COMM_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 293: Configuration for Product CommodityAsianOptionGeometricPrice, Model BlackScholesMerton, Engine AnalyticDiscreteGeomtetricAPEngine

BlackScholesMerton/AnalyticContinuousGeometricAPEngine builds a AnalyticContinuousGeometricAveragePriceAsianEngine. A sample configuration is shown in listing 294.

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="CommodityAsianOptionGeometricPrice">
    <Model>BlackScholesMerton</Model>
    <ModelParameters/>
    <Engine>AnalyticContinuousGeometricAPEngine</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">COMM_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 294: Configuration for Product CommodityAsianOptionGeometricPrice, Model BlackScholesMerton, Engine AnalyticContinuousGeomtetricAPEngine

ScriptedTrade/ScriptedTrade delegates to the scripted trade engine and the associated pricing engine configuration for Product Type ScriptedTrade, see there for details. A sample configuration is given in listing 295.

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="CommodityAsianOptionGeometricPrice">
    <Model>ScriptedTrade</Model>
    <ModelParameters/>
    <Engine>ScriptedTrade</Engine>
```

```
    <EngineParameters>
        <Parameter name="SensitivityTemplate">COMM_MC</Parameter>
    </EngineParameters>
</Product>
```

Listing 295: Configuration for Product CommodityAsianOptionGeometricPrice, Model ScriptedTrade, Engine ScriptedTrade

9.40 Product Type: CommodityAsianOptionGeometricStrike

Used by trade type: CommodityAsianOption if payoffType2 is Geometric and payoffType is AverageStrike

Available Model/Engine pairs:

- BlackScholesMerton/MCDiscreteGeometricASEngine
- BlackScholesMerton/AnalyticDiscreteGeometricASEngine
- ScriptedTrade/ScriptedTrade

Engine description:

BlackScholesMerton/MCDiscreteGeometricASEngine builds a MCDiscreteGeometricASEngine using Sobol sequences. A sample configuration is shown in listing 296.

The parameters have the following meaning:

- BrownianBridge: whether to use Brownian Bridge for MC simulation
- AntitheticVariate: whether to use antithetic variates for MC simulation
- ControlVariate: whether to use control variate for MC simulation
- RequiredSamples: minimum number of samples for MC simulation, if 0 (not specified) RequiredTolerance must be given
- RequiredTolernace: max tolerance for MC error, if 0 (not specified) RequiredSamples must be given
- MaxSamples: max number of samples for MC simulation
- Seed: seed for random number generator
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="CommodityAsianOptionGeometricStrike">
    <Model>BlackScholesMerton</Model>
    <ModelParameters/>
    <Engine>MCDiscreteGeometricASEngine</Engine>
    <EngineParameters>
        <Parameter name="BrownianBridge">true</Parameter>
        <Parameter name="AntitheticVariate">true</Parameter>
        <Parameter name="ControlVariate">true</Parameter>
        <Parameter name="RequiredSamples">10000</Parameter>
        <Parameter name="RequiredTolerance">0</Parameter>
        <Parameter name="MaxSamples">0</Parameter>
        <Parameter name="Seed">42</Parameter>
```

```
        <Parameter name="SensitivityTemplate">COMM_MC</Parameter>
    </EngineParameters>
</Product>
```

Listing 296: Configuration for Product CommodityAsianOptionGeometricStrike, Model BlackScholesMerton, Engine MCDiscreteGeometricASEngine

BlackScholesMerton/AnalyticDiscreteGeometricASEngine builds a AnalyticDiscreteGeometricAverageStrikeAsianEngine. A sample configuration is shown in listing 297.

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="CommodityAsianOptionGeometricStrike">
    <Model>BlackScholesMerton</Model>
    <ModelParameters/>
    <Engine>AnalyticDiscreteGeometricASEngine</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">COMM_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 297: Configuration for Product CommodityAsianOptionGeometricStrike, Model BlackScholesMerton, Engine AnalyticDiscreteGeomtetricASEngine

ScriptedTrade/ScriptedTrade delegates to the scripted trade engine and the associated pricing engine configuration for Product Type ScriptedTrade, see there for details. A sample configuration is given in listing 298.

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="CommodityAsianOptionGeometricStrike">
    <Model>ScriptedTrade</Model>
    <ModelParameters/>
    <Engine>ScriptedTrade</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">COMM_MC</Parameter>
    </EngineParameters>
</Product>
```

Listing 298: Configuration for Product CommodityAsianOptionGeometricStrike, Model ScriptedTrade, Engine ScriptedTrade

9.41 Product Type: CommoditySpreadOption

Used by trade type: CommoditySpreadOption

Available Model/Engine pairs: BlackScholes/CommoditySpreadOptionEngine

Engine description:

BlackScholes/CommoditySpreadOptionEngine builds a CommoditySpreadOptionAnalyticalEngine, which uses the Kirk approximation

described in Iain J. Clark, Commodity Option Pricing, Section 2.9. The correlation between two commodities resp. the intra-asset correlation between two future contracts is parametrized as

$$\rho(s,t) = e^{-\beta|s-t|}$$

where s and t are times to futures expiry. A sample configuration is shown in listing 299.

The parameters have the following meaning:

- beta: the parameter "beta" in the correlation parametrization
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="CommoditySpreadOption">
    <Model>BlackScholes</Model>
    <ModelParameters/>
    <Engine>CommoditySpreadOptionEngine</Engine>
    <EngineParameters>
        <Parameter name="beta">2.05</Parameter>
        <Parameter name="SensitivityTemplate">COMM_Analytical</Parameter>
    </EngineParameters>
    </Product>
```

Listing 299: Configuration for Product CommoditySpreadOption, Model BlackScholes, Engine CommoditySpreadOptionEngine

9.42 Product Type: CommodityOption

Used by trade type: CommodityOption

Available Model/Engine pairs: BlackScholes/AnalyticEuropeanEngine

Engine description:

BlackScholes/AnalyticEuropeanEngine builds a AnalyticEuropeanEngine. A sample configuration is shown in listing 300

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="CommodityOption">
    <Model>BlackScholes</Model>
    <ModelParameters/>
    <Engine>AnalyticEuropeanEngine</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">COMM_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 300: Configuration for Product CommodityOption, Model BlackScholes, Engine AnalyticEuropeanEngine

9.43 Product Type: CommodityOptionForward

Used by trade type: CommodityOption if a future is referenced

Available Model/Engine pairs: BlackScholes/AnalyticEuropeanForwardEngine

Engine description:

BlackScholes/AnalyticEuropeanForwardEngine builds a AnalyticEuropeanForwardEngine. A sample configuration is shown in listing 301

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="CommodityOptionForward">
    <Model>BlackScholes</Model>
    <ModelParameters/>
    <Engine>AnalyticForwardEuropeanEngine</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">COMM_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 301: Configuration for Product CommodityOptionForward, Model BlackScholes, Engine AnalyticForwardEuropeanEngine

9.44 Product Type: CommodityOptionEuropeanCS

Used by trade type: CommodityOption if cash settled

Available Model/Engine pairs: BlackScholes/AnalyticCashSettledEuropeanEngine

Engine description:

BlackScholes/AnalyticCashSettledEuropeanEngine builds a AnalyticCashSettledEuropeanEngine. A sample configuration is shown in listing 302

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="CommodityOptionEuropeanCS">
    <Model>BlackScholes</Model>
    <ModelParameters/>
    <Engine>AnalyticCashSettledEuropeanEngine</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">COMM_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 302: Configuration for Product CommodityOptionEuropeanCS, Model BlackScholes, Engine AnalyticCashSettledEuropeanEngine

9.45 Product Type: CommodityOptionAmerican

Used by trade type: CommodityOption if exercise style is american

Available Model/Engine pairs:

- BlackScholes/FdBlackScholesVanillaEngine
- BlackScholes/BaroneAdesiWhaleyApproximationEngine

Engine description:

BlackScholes/FdBlackScholesVanillaEngine builds a FdBlackScholesVanillaEngine. A sample configuration is shown in listing 303

The parameters have the following meaning:

- Scheme: The finite difference scheme to use
- TimeStepsPerYear: Time grid specification
- XGrid: State grid specification
- DampingSteps: Number of damping steps taken by FD solver
- EnforceMonotoneVariance [optional]: If true variance is modified to be monotone if needed, defaults to true
- TimeGridMinimumSize [optional]: Minimum number in resulting time grid, defaults to 1 if not given
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="CommodityOptionAmerican">
    <Model>BlackScholes</Model>
    <ModelParameters/>
    <Engine>FdBlackScholesVanillaEngine</Engine>
    <EngineParameters>
        <Parameter name="Scheme">Douglas</Parameter>
        <Parameter name="TimeGridPerYear">100</Parameter>
        <Parameter name="XGrid">100</Parameter>
        <Parameter name="DampingSteps">0</Parameter>
        <Parameter name="EnforceMonotoneVariance">true</Parameter>
        <Parameter name="SensitivityTemplate">COMM_FD</Parameter>
    </EngineParameters>
</Product>
```

Listing 303: Configuration for Product CommodityOptionAmerican, Model BlackScholes, Engine FdBlackScholesVanillaEngine

BlackScholes/BaroneAdesiWhaleyApproximationEngine builds a BaroneAdesiWhaleyApproximationEngine. A sample configuration is shown in listing 304

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="CommodityOptionAmerican">
    <Model>BlackScholes</Model>
    <ModelParameters/>
    <Engine>BaroneAdesiWhaleyApproximationEngine</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">COMM_Analytical</Parameter>
```

```
    </EngineParameters>
</Product>
```

Listing 304: Configuration for Product CommodityOptionAmerican, Model BlackScholes, Engine BaroneAdesiWhaleyApproximationEngine

9.46 Product Type: CommoditySwap

Used by trade type: CommoditySwap

Available Model/Engine pairs: DiscountedCashflows/CommoditySwapEngine

Engine description:

DiscountedCashflows/CommoditySwapEngine builds a DiscountingSwapEngine. A sample configuration is shown in listing 305

The parameters have the following meaning:

-
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="CommoditySwap">
    <Model>DiscountedCashflows</Model>
    <ModelParameters/>
    <Engine>CommoditySwapEngine</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">COMM_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 305: Configuration for Product CommoditySwap, Model DiscountedCashflows, Engine CommoditySwapEngine

9.47 Product Type: CommoditySwaption

Used by trade type: CommoditySwaption

Available Model/Engine pairs:

- Black/AnalyticalApproximation
- Black/MonteCarlo

Engine description:

Black/AnalyticalApproximation builds a CommoditySwaptionEngine using moment matching. The correlation between two future contracts is parametrized as

$$\rho(s,t) = e^{-\beta|s-t|}$$

where s and t are times to futures expiry. A sample configuration is shown in listing 306

The parameters have the following meaning:

- beta: the parameter “beta” in the correlation parametrization
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="CommoditySwaption">
    <Model>Black</Model>
    <ModelParameters/>
    <Engine>AnalyticalApproximation</Engine>
    <EngineParameters>
        <Parameter name="beta">2.05</Parameter>
        <Parameter name="SensitivityTemplate">COMM_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 306: Configuration for Product CommoditySwaption, Model Black, Engine AnalyticalApproximation

Black/MonteCarlo builds a CommoditySwaptionMonteCarloEngine using Sobol sequences. The same correlation parametrization as above is used. A sample configuration is shown in listing 307

The parameters have the following meaning:

- beta: the parameter “beta” in the correlation parametrization
- samples: the number of MC samples to use
- seed: the seed to use
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="CommoditySwaption">
    <Model>Black</Model>
    <ModelParameters/>
    <Engine>MonteCarlo</Engine>
    <EngineParameters>
        <Parameter name="beta">2.05</Parameter>
        <Parameter name="samples">10000</Parameter>
        <Parameter name="seed">42</Parameter>
        <Parameter name="SensitivityTemplate">COMM_MC</Parameter>
    </EngineParameters>
</Product>
```

Listing 307: Configuration for Product CommoditySwaption, Model Black, Engine MonteCarlo

9.48 Product Type: EquityAsianOptionArithmeticPrice

Used by trade type: EquityAsianOption if payoffType2 is Arithmetric and payoffType is Asian

Available Model/Engine pairs:

- BlackScholesMerton/MCDiscreteArithmeticAPEngine
- BlackScholesMerton/TurnbullWakemanAsianEngine
- ScriptedTrade/ScriptedTrade

Engine description:

BlackScholesMerton/MCDiscreteArithmeticAPEngine builds a MCDiscreteArithmeticAPEngine using Sobol sequences. A sample configuration is shown in listing 308.

The parameters have the following meaning:

- BrownianBridge: whether to use Brownian Bridge for MC simulation
- AntitheticVariate: whether to use antithetic variates for MC simulation
- ControlVariate: whether to use control variate for MC simulation
- RequiredSamples: minimum number of samples for MC simulation, if 0 (not specified) RequiredTolerance must be given
- RequiredTolernace: max tolerance for MC error, if 0 (not specified) RequiredSamples must be given
- MaxSamples: max number of samples for MC simulation
- Seed: seed for random number generator
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="EquityAsianOptionArithmeticPrice">
    <Model>BlackScholesMerton</Model>
    <ModelParameters/>
    <Engine>MCDiscreteArithmeticAPEngine</Engine>
    <EngineParameters>
        <Parameter name="BrownianBridge">true</Parameter>
        <Parameter name="AntitheticVariate">true</Parameter>
        <Parameter name="ControlVariate">true</Parameter>
        <Parameter name="RequiredSamples">10000</Parameter>
        <Parameter name="RequiredTolerance">0</Parameter>
        <Parameter name="MaxSamples">0</Parameter>
        <Parameter name="Seed">42</Parameter>
        <Parameter name="SensitivityTemplate">EQ_MC</Parameter>
    </EngineParameters>
</Product>
```

Listing 308: Configuration for Product EquityAsianOptionArithmeticPrice, Model BlackScholesMerton, Engine MCDiscreteArithmeticAPEngine

BlackScholesMerton/TurnbullWakemanAsianEngine builds a TurnbullWakemanAsianEngine. A sample configuration is shown in listing 309.

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="EquityAsianOptionArithmeticPrice">
    <Model>BlackScholesMerton</Model>
    <ModelParameters/>
    <Engine>TurnbullWakemanAsianEngine</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">EQ_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 309: Configuration for Product EquityAsianOptionArithmeticPrice, Model BlackScholesMerton, Engine TurnbullWakemanAsianEngine

ScriptedTrade/ScriptedTrade delegates to the scripted trade engine and the associated pricing engine configuration for Product Type ScriptedTrade, see there for details. A sample configuration is given in listing 310.

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="EquityAsianOptionArithmeticPrice">
    <Model>ScriptedTrade</Model>
    <ModelParameters/>
    <Engine>ScriptedTrade</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">EQ_MC</Parameter>
    </EngineParameters>
</Product>
```

Listing 310: Configuration for Product EquityAsianOptionArithmeticPrice, Model ScriptedTrade, Engine ScriptedTrade

9.49 Product Type: EquityAsianOptionArithmeticStrike

Used by trade type: EquityAsianOption if payoffType2 is Arithmetric and payoffType is AverageStrike

Available Model/Engine pairs:

- BlackScholesMerton/MCDiscreteArithmeticASEngine
- ScriptedTrade/ScriptedTrade

Engine description:

BlackScholesMerton/MCDiscreteArithmeticASEngine builds a MCDiscreteArithmeticASEngine using Sobol sequences. A sample configuration is shown in listing 311.

The parameters have the following meaning:

- BrownianBridge: whether to use Brownian Bridge for MC simulation
- AntitheticVariate: whether to use antithetic variates for MC simulation
- ControlVariate: whether to use control variate for MC simulation
- RequiredSamples: minimum number of samples for MC simulation, if 0 (not specified) RequiredTolerance must be given
- RequiredTolernace: max tolerance for MC error, if 0 (not specified) RequiredSamples must be given
- MaxSamples: max number of samples for MC simulation
- Seed: seed for random number generator

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="EquityAsianOptionArithmeticStrike">
    <Model>BlackScholesMerton</Model>
    <ModelParameters/>
    <Engine>MCDiscreteArithmeticASEngine</Engine>
    <EngineParameters>
        <Parameter name="BrownianBridge">true</Parameter>
        <Parameter name="AntitheticVariate">true</Parameter>
        <Parameter name="ControlVariate">true</Parameter>
        <Parameter name="RequiredSamples">10000</Parameter>
        <Parameter name="RequiredTolerance">0</Parameter>
        <Parameter name="MaxSamples">0</Parameter>
        <Parameter name="Seed">42</Parameter>
        <Parameter name="SensitivityTemplate">EQ_MC</Parameter>
    </EngineParameters>
</Product>
```

Listing 311: Configuration for Product EquityAsianOptionArithmeticStrike, Model BlackScholesMerton, Engine MCDiscreteArithmeticASEngine

ScriptedTrade/ScriptedTrade delegates to the scripted trade engine and the associated pricing engine configuration for Product Type ScriptedTrade, see there for details. A sample configuration is given in listing 312.

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="EquityAsianOptionArithmeticStrike">
    <Model>ScriptedTrade</Model>
    <ModelParameters/>
    <Engine>ScriptedTrade</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">EQ_MC</Parameter>
    </EngineParameters>
</Product>
```

Listing 312: Configuration for Product EquityAsianOptionArithmeticStrike, Model ScriptedTrade, Engine ScriptedTrade

9.50 Product Type: EquityAsianOptionGeometricPrice

Used by trade type: EquityAsianOption if payoffType2 is Geometric and payoffType is Asian

Available Model/Engine pairs:

- BlackScholesMerton/MCDiscreteGeometricAPEngine
- BlackScholesMerton/AnalyticDiscreteGeometricAPEngine
- BlackScholesMerton/AnalyticContinuousGeometricAPEngine
- ScriptedTrade/ScriptedTrade

Engine description:

BlackScholesMerton/MCDiscreteGeometricAPEngine builds a MCDiscreteGeometricAPEngine using Sobol sequences. A sample configuration is shown in listing 313.

The parameters have the following meaning:

- BrownianBridge: whether to use Brownian Bridge for MC simulation
- AntitheticVariate: whether to use antithetic variates for MC simulation
- ControlVariate: whether to use control variate for MC simulation
- RequiredSamples: minimum number of samples for MC simulation, if 0 (not specified) RequiredTolerance must be given
- RequiredTolernace: max tolerance for MC error, if 0 (not specified) RequiredSamples must be given
- MaxSamples: max number of samples for MC simulation
- Seed: seed for random number generator
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="EquityAsianOptionGeometricPrice">
    <Model>BlackScholesMerton</Model>
    <ModelParameters/>
    <Engine>MCDiscreteGeometricAPEngine</Engine>
    <EngineParameters>
        <Parameter name="BrownianBridge">true</Parameter>
        <Parameter name="AntitheticVariate">true</Parameter>
        <Parameter name="ControlVariate">true</Parameter>
        <Parameter name="RequiredSamples">10000</Parameter>
        <Parameter name="RequiredTolerance">0</Parameter>
        <Parameter name="MaxSamples">0</Parameter>
        <Parameter name="Seed">42</Parameter>
        <Parameter name="SensitivityTemplate">EQ_MC</Parameter>
    </EngineParameters>
</Product>
```

Listing 313: Configuration for Product EquityAsianOptionGeometricPrice, Model BlackScholesMerton, Engine MCDiscreteGeometricAPEngine

BlackScholesMerton/AnalyticDiscreteGeometricAPEngine builds a AnalyticDiscreteGeometricAveragePriceAsianEngine. A sample configuration is shown in listing 314.

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="EquityAsianOptionGeometricPrice">
    <Model>BlackScholesMerton</Model>
    <ModelParameters/>
    <Engine>AnalyticDiscreteGeometricAPEngine</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">EQ_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 314: Configuration for Product EquityAsianOptionGeometricPrice, Model BlackScholesMerton, Engine AnalyticDiscreteGeomtetricAPEngine

BlackScholesMerton/AnalyticContinuousGeometricAPEngine builds a AnalyticContinuousGeometricAveragePriceAsianEngine. A sample configuration is shown in listing 315.

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="EquityAsianOptionGeometricPrice">
    <Model>BlackScholesMerton</Model>
    <ModelParameters/>
    <Engine>AnalyticContinuousGeometricAPEngine</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">EQ_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 315: Configuration for Product EquityAsianOptionGeometricPrice, Model BlackScholesMerton, Engine AnalyticContinuousGeomtetricAPEngine

ScriptedTrade/ScriptedTrade delegates to the scripted trade engine and the associated pricing engine configuration for Product Type ScriptedTrade, see there for details. A sample configuration is given in listing 316.

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="EquityAsianOptionGeometricPrice">
    <Model>ScriptedTrade</Model>
    <ModelParameters/>
    <Engine>ScriptedTrade</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">EQ_MC</Parameter>
    </EngineParameters>
</Product>
```

Listing 316: Configuration for Product EquityAsianOptionGeometricPrice, Model ScriptedTrade, Engine ScriptedTrade

9.51 Product Type: EquityAsianOptionGeometricStrike

Used by trade type: EquityAsianOption if payoffType2 is Geometric and payoffType is AverageStrike

Available Model/Engine pairs:

- BlackScholesMerton/MCDiscreteGeometricASEngine
- BlackScholesMerton/AnalyticDiscreteGeometricASEngine
- ScriptedTrade/ScriptedTrade

Engine description:

BlackScholesMerton/MCDiscreteGeometricASEngine builds a MCDiscreteGeometricASEngine using Sobol sequences. A sample configuration is shown in listing 317.

The parameters have the following meaning:

- BrownianBridge: whether to use Brownian Bridge for MC simulation
- AntitheticVariate: whether to use antithetic variates for MC simulation
- ControlVariate: whether to use control variate for MC simulation
- RequiredSamples: minimum number of samples for MC simulation, if 0 (not specified) RequiredTolerance must be given
- RequiredTolernace: max tolerance for MC error, if 0 (not specified) RequiredSamples must be given
- MaxSamples: max number of samples for MC simulation
- Seed: seed for random number generator
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="EquityAsianOptionGeometricStrike">
    <Model>BlackScholesMerton</Model>
    <ModelParameters/>
    <Engine>MCDiscreteGeometricASEngine</Engine>
    <EngineParameters>
        <Parameter name="BrownianBridge">true</Parameter>
        <Parameter name="AntitheticVariate">true</Parameter>
        <Parameter name="ControlVariate">true</Parameter>
        <Parameter name="RequiredSamples">10000</Parameter>
        <Parameter name="RequiredTolerance">0</Parameter>
        <Parameter name="MaxSamples">0</Parameter>
        <Parameter name="Seed">42</Parameter>
        <Parameter name="SensitivityTemplate">EQ_MC</Parameter>
    </EngineParameters>
</Product>
```

Listing 317: Configuration for Product EquityAsianOptionGeometricStrike, Model BlackScholesMerton, Engine MCDiscreteGeometricASEngine

BlackScholesMerton/AnalyticDiscreteGeometricASEngine builds a AnalyticDiscreteGeometricAverageStrikeAsianEngine. A sample configuration is shown in listing 318.

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="EquityAsianOptionGeometricStrike">
    <Model>BlackScholesMerton</Model>
    <ModelParameters/>
    <Engine>AnalyticDiscreteGeometricASEngine</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">EQ_Analytical</Parameter>
```

```
    </EngineParameters>
</Product>
```

Listing 318: Configuration for Product EquityAsianOptionGeometricStrike, Model BlackScholesMerton, Engine AnalyticDiscreteGeomtetricASEngine

ScriptedTrade/ScriptedTrade delegates to the scripted trade engine and the associated pricing engine configuration for Product Type ScriptedTrade, see there for details. A sample configuration is given in listing 319.

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="EquityAsianOptionGeometricStrike">
    <Model>ScriptedTrade</Model>
    <ModelParameters/>
    <Engine>ScriptedTrade</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">EQ_MC</Parameter>
    </EngineParameters>
</Product>
```

Listing 319: Configuration for Product EquityAsianOptionGeometricStrike, Model ScriptedTrade, Engine ScriptedTrade

9.52 Product Type: EquityBarrierOption, FxBarrierOption

Used by trade type: EquityBarrierOption resp. FxBarrierOption

Available Model/Engine pairs:

- BlackScholesMerton/AnalyticBarrierEngine
- BlackScholesMerton/FdBlackScholesBarrierEngine

Engine description:

BlackScholes/AnalyticBarrierEngine builds a AnalyticBarrierEngine. A sample configuration is shown in listing 320 for equity. The configuration for fx is identical except the Model is set to GarmanKohlhagen.

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="EquityBarrierOption">
    <Model>BlackScholesMerton</Model>
    <ModelParameters/>
    <Engine>AnalyticBarrierEngine</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">EQ_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 320: Configuration for Product EquityBarrierOption, Model BlackScholesMertong Engine AnalyticBarrierEngine

BlackScholes/FdBlackScholesBarrierEngine builds a FdBlackScholesBarrierEngine. A sample configuration is shown in listing 321 for equity. The configuration for fx is identical except the Model is set to GarmanKohlhagen.

The parameters have the following meaning:

- Scheme: The finite difference scheme to use
- TimeStepsPerYear: Time grid specification
- XGrid: State grid specification
- DampingSteps: Number of damping steps taken by FD solver
- EnforceMonotoneVariance [optional]: If true variance is modified to be monotone if needed, defaults to true
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="EquityBarrierOption">
    <Model>BlackScholesMerton</Model>
    <ModelParameters/>
    <Engine>FdBlackScholesBarrierEngine</Engine>
    <EngineParameters>
        <Parameter name="Scheme">Douglas</Parameter>
        <Parameter name="TimeGridPerYear">100</Parameter>
        <Parameter name="XGrid">100</Parameter>
        <Parameter name="DampingSteps">0</Parameter>
        <Parameter name="EnforceMonotoneVariance">true</Parameter>
        <Parameter name="SensitivityTemplate">EQ_FD</Parameter>
    </EngineParameters>
</Product>
```

Listing 321: Configuration for Product EquityBarrierOption, Model BlackScholesMerton, Engine FdBlackScholesBarrierEngine

9.53 Product Type: EquityDoubleBarrierOption, FxDoubleBarrierOption

Used by trade type: EquityDoubleBarrierOption, FxDoubleBarrierOption

Available Model/Engine pairs: GarmanKohlhagen/AnalyticDoubleBarrierEngine (both fx and equity)

Engine description:

GarmanKohlhagen/AnalyticDoubleBarrierEngine builds a AnalyticDoubleBarrierEngine. A sample configuration is shown in listing 322. The configuration for equity is identical.

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="FxDoubleBarrierOption">
    <Model>GarmanKohlhagen</Model>
    <ModelParameters/>
    <Engine>AnalyticDoubleBarrierEngine</Engine>
```

```
    <EngineParameters>
        <Parameter name="SensitivityTemplate">FX_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 322: Configuration for Product FxDoubleBarrierOption, Model GarmanKohlhagen, Engine AnalyticDoubleBarrierBinaryEngine

9.54 Product Type: EquityDigitalOption, FxDigitalOption

Used by trade type: EquityDigitalOption, FxDigitalOption

Available Model/Engine pairs: BlackScholesMerton/AnalyticEuropeanEngine (equity) resp. GarmanKohlhagen/AnalyticEuropeanEngine (fx)

Engine description:

BlackScholesMerton/AnalyticEuropeanEngine (equity) resp. GarmanKohlhagen/AnalyticEuropeanEngine (fx) builds a AnalyticEuropeanEngine. A sample configuration is shown in listing 323 for fx. The configuration for equity is identical except the Model is set to BlackScholesMerton.

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
 <Product type="FxDigitalOption">
    <Model>GarmanKohlhagen</Model>
    <ModelParameters/>
    <Engine>AnalyticEuropeanEngine</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">FX_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 323: Configuration for Product FxDigitalOption, Model GarmanKohlhagen, Engine AnalyticEuropeanEngine

9.55 Product Type: EquityEuropeanCompositeOption

Used by trade type: EquityOption if a composite option is built

Available Model/Engine pairs: BlackScholes/AnalyticEuropeanEngine

Engine description:

BlackScholes/AnalyticEuropeanEngine builds a AnalyticEuropeanEngine (with a vol composed from the relevant equity and fx vol surfaces). A sample configuration is shown in listing 324

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="EquityEuropeanCompositeOption">
    <Model>BlackScholes</Model>
    <ModelParameters/>
```

```
    <Engine>AnalyticEuropeanEngine</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">EQ_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 324: Configuration for Product EquityEuropeanCompositeOption, Model BlackScholes, Engine AnalyticEuropeanEngine

9.56 Product Type: EquityForward

Used by trade type: EquityForward

Available Model/Engine pairs: DiscountedCashflows/DiscountingEquityForwardEngine

Engine description:

DiscountedCashflows/DiscountingEquityForwardEngine builds a DiscountingEquityForwardEngine. A sample configuration is shown in listing 325

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="EquityForward">
    <Model>DiscountedCashflows</Model>
    <ModelParameters/>
    <Engine>DiscountingEquityForwardEngine</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">EQ_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 325: Configuration for Product EquiytForward, Model DiscountedCashflows, Engine DiscountingEquityForwardEngine

9.57 Product Type: EquityFutureOption

Used by trade type: EquityFuturesOption

Available Model/Engine pairs: BlackScholes/AnalyticEuroepanForwardEngine

Engine description:

BlackScholes/AnalyticEuroepanForwardEngine builds a AnalyticEuropeanForwardEngine. A sample configuration is shown in listing 326

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="EquityFutureOption">
  <Model>BlackScholes</Model>
  <ModelParameters />
  <Engine>AnalyticEuropeanForwardEngine</Engine>
  <EngineParameters>
      <Parameter name="SensitivityTemplate">COMM_Analytical</Parameter>
```

```
  </EngineParameters>
</Product>
```

Listing 326: Configuration for Product EquiytFutureOption, Model BlackScholes, Engine AnalyticEuropeanForwardEngine

9.58 Product Type: EquityOption

Used by trade type: EquityOption

Available Model/Engine pairs: BlackScholesMerton/AnalyticEuropeanEngine

Engine description:

BlackScholesMerton/AnalyticEuropeanEngine builds a AnalyticEuropeanEngine. A sample configuration is shown in listing 327

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="EquityOption">
    <Model>BlackScholesMerton</Model>
    <ModelParameters/>
    <Engine>AnalyticEuropeanEngine</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">COMM_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 327: Configuration for Product EquityOption, Model BlackScholesMerton, Engine AnalyticEuropeanEngine

9.59 Product Type: EquityCliquetOption

Used by trade type: EquityCliquetOption

Available Model/Engine pairs: BlackScholes/MCScript

Engine description:

BlackScholes/MCScript builds a CliquetOptionMcScriptEngine which uses the scripted trade framework. A sample configuration is shown in listing 328

The parameters have the following meaning:

- Samples: number of MC samples to use
- RegressionOrder: the regression order to use for conditional expectations (not used by this engine)
- Interactive: whether to enable the interactive debugger
- ScriptedLibraryOverride: whether to override a script 'EquityCliquetOption' in the script library with a hardcoded script
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="EquityCliquetOption">
    <Model>BlackScholes</Model>
    <ModelParameters/>
    <Engine>MCScript</Engine>
    <EngineParameters>
        <Parameter name="Samples">10000</Parameter>
        <Parameter name="RegressionOrder">6</Parameter>
        <Parameter name="Interactive">false</Parameter>
        <Parameter name="ScriptedLibraryOverride">false</Parameter>
        <Parameter name="SensitivityTemplate">EQ_MC</Parameter>
    </EngineParameters>
</Product>
```

Listing 328: Configuration for Product EquityCliquetOption, Model BlackScholes, Engine MCScript

9.60 Product Type: QuantoEquityOption

Used by trade type: EquityOption with quanto payoff (pay currency does not match equity currency)

Available Model/Engine pairs: BlackScholes/AnalyticEuropeanEngine

Engine description:

BlackScholes/AnalyticEuropeanEngine builds a AnalyticEuropeanEngine. A sample configuration is shown in listing 329

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="QuantoEquityOption">
    <Model>BlackScholes</Model>
    <ModelParameters>
        <Parameter name="FXSource">GENERIC</Parameter>
    </ModelParameters>
    <Engine>AnalyticEuropeanEngine</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">EQ_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 329: Configuration for Product QuantoEquityOption, Model BlackScholes, Engine AnalyticEuropeanEngine

9.61 Product Type: EquityOptionEuropeanCS

Used by trade type: EquityOption if cash settled

Available Model/Engine pairs:
BlackScholesMerton/AnalyticCashSettledEuropeanEngine

Engine description:

BlackScholesMerton/AnalyticCashSettledEuropeanEngine builds a AnalyticCashSettledEuropeanEngine. A sample configuration is shown in listing 330

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="EquityOptionEuropeanCS">
    <Model>BlackScholesMerton</Model>
    <ModelParameters/>
    <Engine>AnalyticCashSettledEuropeanEngine</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">COMM_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 330: Configuration for Product EquityOptionEuropeanCS, Model BlackScholesMerton, Engine AnalyticCashSettledEuropeanEngine

9.62 Product Type: EquityOptionAmerican

Used by trade type: EquityOption if exercise style is american

Available Model/Engine pairs:

- BlackScholesMerton/FdBlackScholesVanillaEngine
- BlackScholesMerton/BaroneAdesiWhaleyApproximationEngine

Engine description:

BlackScholesMerton/FdBlackScholesVanillaEngine builds a FdBlackScholesVanillaEngine. A sample configuration is shown in listing 333

The parameters have the following meaning:

- Scheme: The finite difference scheme to use
- TimeStepsPerYear: Time grid specification
- XGrid: State grid specification
- DampingSteps: Number of damping steps taken by FD solver
- EnforceMonotoneVariance [optional]: If true variance is modified to be monotone if needed, defaults to true
- TimeGridMinimumSize [optional]: Minimum number in resulting time grid, defaults to 1 if not given
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="EquityOptionAmerican">
    <Model>BlackScholesMerton</Model>
    <ModelParameters/>
    <Engine>FdBlackScholesVanillaEngine</Engine>
    <EngineParameters>
        <Parameter name="Scheme">Douglas</Parameter>
        <Parameter name="TimeGridPerYear">100</Parameter>
```

```
        <Parameter name="XGrid">100</Parameter>
        <Parameter name="DampingSteps">0</Parameter>
        <Parameter name="EnforceMonotoneVariance">true</Parameter>
        <Parameter name="SensitivityTemplate">COMM_FD</Parameter>
    </EngineParameters>
</Product>
```

Listing 331: Configuration for Product EquityOptionAmerican, Model BlackScholesMerton, Engine FdBlackScholesVanillaEngine

BlackScholesMerton/BaroneAdesiWhaleyApproximationEngine builds a BaroneAdesiWhaleyApproximationEngine. A sample configuration is shown in listing 332

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="EquityOptionAmerican">
    <Model>BlackScholesMerton</Model>
    <ModelParameters/>
    <Engine>BaroneAdesiWhaleyApproximationEngine</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">COMM_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 332: Configuration for Product EquityOptionAmerican, Model BlackScholesMerton, Engine BaroneAdesiWhaleyApproximationEngine

9.63 Product Type: QuantoEquityOptionAmerican

Used by trade type: EquityOption if exercise style is american and the payoff Currency is different than currency the underlying equity is quoted in.

Available Model/Engine pairs:

- BlackScholesMerton/FdBlackScholesVanillaEngine

Engine description:

BlackScholesMerton/FdBlackScholesVanillaEngine builds a FdBlackScholesVanillaEngine. A sample configuration is shown in listing 333

The parameters have the following meaning:

- Scheme: The finite difference scheme to use
- TimeStepsPerYear: Time grid specification
- XGrid: State grid specification
- DampingSteps: Number of damping steps taken by FD solver
- EnforceMonotoneVariance [optional]: If true variance is modified to be monotone if needed, defaults to true

- TimeGridMinimumSize [optional]: Minimum number in resulting time grid, defaults to 1 if not given
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="QuantoEquityOptionAmerican">
    <Model>BlackScholesMerton</Model>
    <ModelParameters/>
    <Engine>FdBlackScholesVanillaEngine</Engine>
    <EngineParameters>
        <Parameter name="Scheme">Douglas</Parameter>
        <Parameter name="TimeGridPerYear">100</Parameter>
        <Parameter name="XGrid">100</Parameter>
        <Parameter name="DampingSteps">0</Parameter>
        <Parameter name="EnforceMonotoneVariance">true</Parameter>
        <Parameter name="SensitivityTemplate">COMM_FD</Parameter>
    </EngineParameters>
</Product>
```

Listing 333: Configuration for Product QuantoEquityOptionAmerican, Model BlackScholesMerton, Engine FdBlackScholesVanillaEngine

9.64 Product Type: EquityTouchOption, FxTouchOption

Used by trade type: EquityTouchOption, FxTouchOption

Available Model/Engine pairs: BlackScholesMerton/AnalyticDigitalAmericanEngine for equity resp. GarmanKohlhagen/AnalyticDigitalAmericanEngine for fx

Engine description:

BlackScholesMerton/AnalyticDigitalAmericanEngine (equity) reps. GarmanKohlhagen/AnalyticDigitalAmericanEngine (fx) builds a AnalyticDigitalAmericanEngine (one touch) or AnalyticDigitalAmericanKOEngine (no touch). A sample configuration is shown in listing 334 for fx. The configuration for equity is the same except the Model ist set to BlackScholesMerton.

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="FxTouchOption">
    <Model>GarmanKohlhagen</Model>
    <ModelParameters/>
    <Engine>AnalyticDigitalAmericanEngine</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">FX_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 334: Configuration for Product FxTouchOption, Model GarmanKohlhagen, Engine AnalyticDigitalAmericanEngine

9.65 Product Type: ForwardBond

Used by trade type: ForwardBond

Available Model/Engine pairs: DiscountedCashflows/DiscountingForwardBondEngine

Engine description:

DiscountedCashflows/DiscountingForwardBondEngine builds a DiscountingForwardBondEngine. A sample configuration is shown in listing 335.

The parameters have the following meaning:

- OpenEndDateReplacement: if end date is missing in bond schedule (perpetual bond), valuation date plus this period is used as a replacement date
- TimestepPeriod: discretization interval for zero bond pricing
- SensitivityTemplate [optional]: the sensitivity template to use
- SpreadOnIncomeCurve [optional]: whether to apply the security spread on the income curve for compounding, default to false

```
<Product type="ForwardBond">
    <Model>DiscountedCashflows</Model>
    <ModelParameters>
      <Parameter name="OpenEndDateReplacement">50Y</Parameter>
    </ModelParameters>
    <Engine>DiscountingForwardBondEngine</Engine>
    <EngineParameters>
        <Parameter name="TimestepPeriod">3M</Parameter>
        <Parameter name="SensitivityTemplate">IR_Analytical</Parameter>
        <Parameter name="SpreadOnIncomeCurve">true</Parameter>
    </EngineParameters>
</Product>
```

Listing 335: Configuration for Product ForwardBond, Model DiscountedCashflows, Engine DiscountingForwardBondEngine

9.66 Product Type: EquityVarianceSwap, CommodityVarianceSwap, FxVarianceSwap

Used by trade type: EquityVarianceSwap, CommodityVarianceSwap, FxVarianceSwap

Available Model/Engine pairs: BlackScholesMerton/EquityVarianceSwap (resp. Commodity, Fx)

Engine description:

BlackScholesMerton/EquityVarianceSwap (resp. Commodity, Fx) builds a GeneralisedReplicatingVarianceSwapEngine or VolatilityFromVarianceSwapEngine depending on MomentType Variance, Volatility. A sample configuration is shown in listing 336 and 337. The example is for Equity, but holds equally for product types CommodityVarianceSwap, FxVarianceSwap.

The parameters have the following meaning:

- Scheme [Optional, default GaussLobatto]: GaussLobatto or Segment, this determines the integration scheme used

- Bounds [Optional, default PriceThreshold]: "Fixed": The integration bounds are found by

$$\text{Lower} = Fe^{m\sigma\sqrt{T}}, \text{Upper} = Fe^{M\sigma\sqrt{T}}$$

  where m is the FixedMinStdDevs and M is the FixedMaxStdDevs parameter listed below and σ is the ATMF volatility of the underlying at maturity T.

  "PriceThreshold": The integration bounds are found by looking for the largest (smallest) strikes Lower, Upper such that the integrand in the replication formula above is below the PriceThreshold parameter listed below. The search starts at the forward level F for Lower, Upper and then Lower, Upper are updated by factors $1-p$ resp. $1+p$ where p is the PriceThresholdStep parameter below, until either the price threshold criterium is met or the number of updates exceeds the MaxPriceThresholdSteps parameter.

- Accuracy [Optional, default 10^{-5}]: Numerical accuracy tolerance for the numerical integration. This only applies to the GaussLobatto scheme.
- MaxIterations [Optional, default 1000]: The maximum number of iterations performed in the numerical integration. This only applies to the GaussLobatto scheme.
- Steps [Optional, default 100]: The number of steps in the Segment numerical integration scheme (only applies for this scheme).
- PriceThreshold [Optional, default 10^{-10}]: Used to determine the integration bounds if Bounds = PriceThredshold, see above.
- MaxPriceThresholdSteps [Optional, default 100]: Used to determine the integration bounds if Bounds = PriceThredshold, see above.
- PriceThresholdStep [Option, default 0.1]: Used to determine the integration bounds if Bounds = PriceThredshold, see above.
- FixedMinStdDevs [Optional, default -5]: Used to determine the integration bounds if Bounds = Fixed, see above.
- FixedMaxStdDevs [Optional, default 5]: Used to determine the integration bounds, if Bounds = Fixed, see above.
- StaticTodaysSpot [Optional, default false]: If true the contribution to the variance from the last day before the valuation date to the valuation date is ignored in scenario / sensitivity calculations. See below for more details.
- SensitivityTemplate [optional]: the sensitivity template to use

9.67 Product Type: FxAsianOptionArithmeticPrice

Used by trade type: FxAsianOption if payoffType2 is Arithmetric and payoffType is Asian

Available Model/Engine pairs:

- BlackScholesMerton/MCDiscreteArithmeticAPEngine

Listing 336: Configuration for Product EquityVarianceSwap, Model BlackScholesMerton, Engine ReplicatingVarianceSwapEngine, "Robust" configuration using PriceThreshold if market smiles are of lower quality.

```
<Product type="EquityVarianceSwap">
  <Model>BlackScholesMerton</Model>
  <ModelParameters>
    <Parameter name="StaticTodaysSpot">false</Parameter>
  </ModelParameters>
  <Engine>ReplicatingVarianceSwapEngine</Engine>
  <EngineParameters>
    <Parameter name="Scheme">Segment</Parameter>
    <Parameter name="Bounds">PriceThreshold</Parameter>
    <Parameter name="Steps">1000</Parameter>
    <Parameter name="PriceThreshold">1E-10</Parameter>
    <Parameter name="MaxPriceThresholdSteps">500</Parameter>
    <Parameter name="PriceThresholdStep">0.1</Parameter>
  </EngineParameters>
</Product>
```

Listing 337: Configuration for Product EquityVarianceSwap, Model BlackScholesMerton, Engine ReplicatingVarianceSwapEngine, Alternative "Robust" variance swap pricing engine configuration using fixed integration bounds.

```
<Product type="EquityVarianceSwap">
  <Model>BlackScholesMerton</Model>
  <ModelParameters>
    <Parameter name="StaticTodaysSpot">false</Parameter>
  </ModelParameters>
  <Engine>ReplicatingVarianceSwapEngine</Engine>
  <EngineParameters>
    <Parameter name="Scheme">Segment</Parameter>
    <Parameter name="Bounds">Fixed</Parameter>
    <Parameter name="Steps">1000</Parameter>
    <Parameter name="FixedMinStdDevs">-5.0</Parameter>
    <Parameter name="FixedMaxStdDevs">5.0</Parameter>
  </EngineParameters>
</Product>
```

- BlackScholesMerton/TurnbullWakemanAsianEngine
- ScriptedTrade/ScriptedTrade

Engine description:

BlackScholesMerton/MCDiscreteArithmeticAPEngine builds a MCDiscreteArithmeticAPEngine using Sobol sequences. A sample configuration is shown in listing 338.

The parameters have the following meaning:

- BrownianBridge: whether to use Brownian Bridge for MC simulation
- AntitheticVariate: whether to use antithetic variates for MC simulation
- ControlVariate: whether to use control variate for MC simulation

- RequiredSamples: minimum number of samples for MC simulation, if 0 (not specified) RequiredTolerance must be given
- RequiredTolernace: max tolerance for MC error, if 0 (not specified) RequiredSamples must be given
- MaxSamples: max number of samples for MC simulation
- Seed: seed for random number generator
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="FxAsianOptionArithmeticPrice">
    <Model>BlackScholesMerton</Model>
    <ModelParameters/>
    <Engine>MCDiscreteArithmeticAPEngine</Engine>
    <EngineParameters>
        <Parameter name="BrownianBridge">true</Parameter>
        <Parameter name="AntitheticVariate">true</Parameter>
        <Parameter name="ControlVariate">true</Parameter>
        <Parameter name="RequiredSamples">10000</Parameter>
        <Parameter name="RequiredTolerance">0</Parameter>
        <Parameter name="MaxSamples">0</Parameter>
        <Parameter name="Seed">42</Parameter>
        <Parameter name="SensitivityTemplate">FX_MC</Parameter>
    </EngineParameters>
</Product>
```

Listing 338: Configuration for Product FxAsianOptionArithmeticPrice, Model BlackScholesMerton, Engine MCDiscreteArithmeticAPEngine

BlackScholesMerton/TurnbullWakemanAsianEngine builds a TurnbullWakemanAsianEngine. A sample configuration is shown in listing 339.

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="FxAsianOptionArithmeticPrice">
    <Model>BlackScholesMerton</Model>
    <ModelParameters/>
    <Engine>TurnbullWakemanAsianEngine</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">FX_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 339: Configuration for Product FxAsianOptionArithmeticPrice, Model BlackScholesMerton, Engine TurnbullWakemanAsianEngine

ScriptedTrade/ScriptedTrade delegates to the scripted trade engine and the associated pricing engine configuration for Product Type ScriptedTrade, see there for details. A sample configuration is given in listing 340.

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="FxAsianOptionArithmeticPrice">
    <Model>ScriptedTrade</Model>
    <ModelParameters/>
    <Engine>ScriptedTrade</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">FX_MC</Parameter>
    </EngineParameters>
</Product>
```

Listing 340: Configuration for Product FxAsianOptionArithmeticPrice, Model ScriptedTrade, Engine ScriptedTrade

9.68 Product Type: FxAsianOptionArithmeticStrike

Used by trade type: FxAsianOption if payoffType2 is Arithmetric and payoffType is AverageStrike

Available Model/Engine pairs:

- BlackScholesMerton/MCDiscreteArithmeticASEngine
- ScriptedTrade/ScriptedTrade

Engine description:

BlackScholesMerton/MCDiscreteArithmeticASEngine builds a MCDiscreteArithmeticASEngine using Sobol sequences. A sample configuration is shown in listing 341.

The parameters have the following meaning:

- BrownianBridge: whether to use Brownian Bridge for MC simulation
- AntitheticVariate: whether to use antithetic variates for MC simulation
- ControlVariate: whether to use control variate for MC simulation
- RequiredSamples: minimum number of samples for MC simulation, if 0 (not specified) RequiredTolerance must be given
- RequiredTolernace: max tolerance for MC error, if 0 (not specified) RequiredSamples must be given
- MaxSamples: max number of samples for MC simulation
- Seed: seed for random number generator
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="FxAsianOptionArithmeticStrike">
    <Model>BlackScholesMerton</Model>
    <ModelParameters/>
    <Engine>MCDiscreteArithmeticASEngine</Engine>
    <EngineParameters>
        <Parameter name="BrownianBridge">true</Parameter>
        <Parameter name="AntitheticVariate">true</Parameter>
        <Parameter name="ControlVariate">true</Parameter>
        <Parameter name="RequiredSamples">10000</Parameter>
```

```
        <Parameter name="RequiredTolerance">0</Parameter>
        <Parameter name="MaxSamples">0</Parameter>
        <Parameter name="Seed">42</Parameter>
        <Parameter name="SensitivityTemplate">FX_MC</Parameter>
    </EngineParameters>
</Product>
```

Listing 341: Configuration for Product FxAsianOptionArithmeticStrike, Model BlackScholesMerton, Engine MCDiscreteArithmeticASEngine

ScriptedTrade/ScriptedTrade delegates to the scripted trade engine and the associated pricing engine configuration for Product Type ScriptedTrade, see there for details. A sample configuration is given in listing 342.

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="FxAsianOptionArithmeticStrike">
    <Model>ScriptedTrade</Model>
    <ModelParameters/>
    <Engine>ScriptedTrade</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">FX_MC</Parameter>
    </EngineParameters>
</Product>
```

Listing 342: Configuration for Product FxAsianOptionArithmeticStrike, Model ScriptedTrade, Engine ScriptedTrade

9.69 Product Type: FxAsianOptionGeometricPrice

Used by trade type: FxAsianOption if payoffType2 is Geometric and payoffType is Asian

Available Model/Engine pairs:

- BlackScholesMerton/MCDiscreteGeometricAPEngine
- BlackScholesMerton/AnalyticDiscreteGeometricAPEngine
- BlackScholesMerton/AnalyticContinuousGeometricAPEngine
- ScriptedTrade/ScriptedTrade

Engine description:

BlackScholesMerton/MCDiscreteGeometricAPEngine builds a MCDiscreteGeometricAPEngine using Sobol sequences. A sample configuration is shown in listing 343.

The parameters have the following meaning:

- BrownianBridge: whether to use Brownian Bridge for MC simulation
- AntitheticVariate: whether to use antithetic variates for MC simulation
- ControlVariate: whether to use control variate for MC simulation

- RequiredSamples: minimum number of samples for MC simulation, if 0 (not specified) RequiredTolerance must be given
- RequiredTolernace: max tolerance for MC error, if 0 (not specified) RequiredSamples must be given
- MaxSamples: max number of samples for MC simulation
- Seed: seed for random number generator
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="FxAsianOptionGeometricPrice">
    <Model>BlackScholesMerton</Model>
    <ModelParameters/>
    <Engine>MCDiscreteGeometricAPEngine</Engine>
    <EngineParameters>
        <Parameter name="BrownianBridge">true</Parameter>
        <Parameter name="AntitheticVariate">true</Parameter>
        <Parameter name="ControlVariate">true</Parameter>
        <Parameter name="RequiredSamples">10000</Parameter>
        <Parameter name="RequiredTolerance">0</Parameter>
        <Parameter name="MaxSamples">0</Parameter>
        <Parameter name="Seed">42</Parameter>
        <Parameter name="SensitivityTemplate">FX_MC</Parameter>
    </EngineParameters>
</Product>
```

Listing 343: Configuration for Product FxAsianOptionGeometricPrice, Model BlackScholesMerton, Engine MCDiscreteGeometricAPEngine

BlackScholesMerton/AnalyticDiscreteGeometricAPEngine builds a AnalyticDiscreteGeometricAveragePriceAsianEngine. A sample configuration is shown in listing 344.

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="FxAsianOptionGeometricPrice">
    <Model>BlackScholesMerton</Model>
    <ModelParameters/>
    <Engine>AnalyticDiscreteGeometricAPEngine</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">FX_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 344: Configuration for Product FxAsianOptionGeometricPrice, Model BlackScholesMerton, Engine AnalyticDiscreteGeomtetricAPEngine

BlackScholesMerton/AnalyticContinuousGeometricAPEngine builds a AnalyticContinuousGeometricAveragePriceAsianEngine. A sample configuration is shown in listing 345.

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="FxAsianOptionGeometricPrice">
    <Model>BlackScholesMerton</Model>
    <ModelParameters/>
    <Engine>AnalyticContinuousGeometricAPEngine</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">FX_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 345: Configuration for Product FxAsianOptionGeometricPrice, Model BlackScholesMerton, Engine AnalyticContinuousGeomtetricAPEngine

ScriptedTrade/ScriptedTrade delegates to the scripted trade engine and the associated pricing engine configuration for Product Type ScriptedTrade, see there for details. A sample configuration is given in listing 346.

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="FxAsianOptionGeometricPrice">
    <Model>ScriptedTrade</Model>
    <ModelParameters/>
    <Engine>ScriptedTrade</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">FX_MC</Parameter>
    </EngineParameters>
</Product>
```

Listing 346: Configuration for Product FxAsianOptionGeometricPrice, Model ScriptedTrade, Engine ScriptedTrade

9.70 Product Type: FxAsianOptionGeometricStrike

Used by trade type: FxAsianOption if payoffType2 is Geometric and payoffType is AverageStrike

Available Model/Engine pairs:

- BlackScholesMerton/MCDiscreteGeometricASEngine
- BlackScholesMerton/AnalyticDiscreteGeometricASEngine
- ScriptedTrade/ScriptedTrade

Engine description:

BlackScholesMerton/MCDiscreteGeometricASEngine builds a MCDiscreteGeometricASEngine using Sobol sequences. A sample configuration is shown in listing 347.

The parameters have the following meaning:

- BrownianBridge: whether to use Brownian Bridge for MC simulation
- AntitheticVariate: whether to use antithetic variates for MC simulation

- ControlVariate: whether to use control variate for MC simulation
- RequiredSamples: minimum number of samples for MC simulation, if 0 (not specified) RequiredTolerance must be given
- RequiredTolernace: max tolerance for MC error, if 0 (not specified) RequiredSamples must be given
- MaxSamples: max number of samples for MC simulation
- Seed: seed for random number generator
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="FxAsianOptionGeometricStrike">
    <Model>BlackScholesMerton</Model>
    <ModelParameters/>
    <Engine>MCDiscreteGeometricASEngine</Engine>
    <EngineParameters>
        <Parameter name="BrownianBridge">true</Parameter>
        <Parameter name="AntitheticVariate">true</Parameter>
        <Parameter name="ControlVariate">true</Parameter>
        <Parameter name="RequiredSamples">10000</Parameter>
        <Parameter name="RequiredTolerance">0</Parameter>
        <Parameter name="MaxSamples">0</Parameter>
        <Parameter name="Seed">42</Parameter>
        <Parameter name="SensitivityTemplate">FX_MC</Parameter>
    </EngineParameters>
</Product>
```

Listing 347: Configuration for Product FxAsianOptionGeometricStrike, Model BlackScholesMerton, Engine MCDiscreteGeometricASEngine

BlackScholesMerton/AnalyticDiscreteGeometricASEngine builds a AnalyticDiscreteGeometricAverageStrikeAsianEngine. A sample configuration is shown in listing 348.

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="FxAsianOptionGeometricStrike">
    <Model>BlackScholesMerton</Model>
    <ModelParameters/>
    <Engine>AnalyticDiscreteGeometricASEngine</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">FX_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 348: Configuration for Product FxAsianOptionGeometricStrike, Model BlackScholesMerton, Engine AnalyticDiscreteGeomtetricASEngine

ScriptedTrade/ScriptedTrade delegates to the scripted trade engine and the associated pricing engine configuration for Product Type ScriptedTrade, see there for details. A sample configuration is given in listing 349.

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="FxAsianOptionGeometricStrike">
    <Model>ScriptedTrade</Model>
    <ModelParameters/>
    <Engine>ScriptedTrade</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">FX_MC</Parameter>
    </EngineParameters>
</Product>
```

Listing 349: Configuration for Product FxAsianOptionGeometricStrike, Model ScriptedTrade, Engine ScriptedTrade

9.71 Product Type: FxDigitalOptionEuropeanCS

Used by trade type: FxDigitalOption if cash settled, FxEuropeanBarrierOption

Available Model/Engine pairs:
GarmanKohlhagen/AnalyticCashSettledEuropeanEngine

Engine description:

GarmanKohlhagen/AnalyticCashSettledEuropeanEngine builds a AnalyticCashSettledEuropeanEngine. A sample configuration is shown in listing 350

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="FxDigitalOptionEuropeanCS">
    <Model>GarmanKohlhagen</Model>
    <ModelParameters/>
    <Engine>AnalyticCashSettledEuropeanEngine</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">FX_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 350: Configuration for Product FxDigitalOptionEuropeanCS, Model GarmanKohlhagen, Engine AnalyticCashSettledEuropeanEngine

9.72 Product Type: FxDigitalBarrierOption

Used by trade type: FxDigitalBarrierOption

Available Model/Engine pairs: GarmanKohlhagen/FdBlackScholesBarrierEngine

Engine description:

GarmanKohlhagen/FdBlackScholesBarrierEngine builds a FdBlackScholesBarrierEngine. A sample configuration is shown in listing 351

The parameters have the following meaning:

- Scheme: The finite difference scheme to use
- TimeStepsPerYear: Time grid specification

- XGrid: State grid specification
- DampingSteps: Number of damping steps taken by FD solver
- EnforceMonotoneVariance [optional]: If true variance is modified to be monotone if needed, defaults to true
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="FxDigitalBarrierOption">
    <Model>GarmanKohlhagen</Model>
    <ModelParameters/>
    <Engine>FdBlackScholesBarrierEngine</Engine>
    <EngineParameters>
        <Parameter name="Scheme">Douglas</Parameter>
        <Parameter name="TimeGridPerYear">100</Parameter>
        <Parameter name="XGrid">100</Parameter>
        <Parameter name="DampingSteps">0</Parameter>
        <Parameter name="SensitivityTemplate">FX_FD</Parameter>
    </EngineParameters>
</Product>
```

Listing 351: Configuration for Product FxDigitalBarrierOption, Model GarmanKohlhagen, Engine FdBlackScholesBarrierEngine

9.73 Product Type: FormulaBasedCoupon

Used by trade type: leg type FormulaBased (coupon pricer)

Available Model/Engine pairs:

- BrigoMercurio/MC

Engine description:

BrigoMercurio/MC builds a MCGaussianFormulaBasedCouponPricer. A sample configuration is shown in listing 352. We refer to the formula based coupon module documentation for further details.

The parameters have the following meaning:

- FXSource: specifies the FX index tag to be used to look up FX-Ibor or FX-CMS correlations
- Samples: number of MC samples to use
- Sobol: whether to use Sobol numbers for simulation
- SalvagingAlgorithm: whether to make the input correlation matrix positive definite. Compare [20].
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="FormulaBasedCoupon">
    <Model>BrigoMercurio</Model>
    <ModelParameters>
        <Parameter name="FXSource">GENERIC</Parameter>
    </ModelParameters>
    <Engine>MC</Engine>
```

```
    <EngineParameters>
        <Parameter name="Samples">10000</Parameter>
        <Parameter name="Seed">42</Parameter>
        <Parameter name="Sobol">Y</Parameter>
        <Parameter name="SalvagingAlgorithm">Spectral</Parameter>
        <Parameter name="SensitivityTemplate">IR_MC</Parameter>
    </EngineParameters>
</Product>
```

Listing 352: Configuration for Product FormulaBasedCoupon, Model BrigoMercurio, Engine MC

9.74 Product Type: FxForward

Used by trade type: FxForward

Available Model/Engine pairs:

- DiscountedCashflows/DiscountingFxForwardEngine
- CrossAssetModel/AMC (for use in AMC simulations only)

Engine description:

DiscountedCashflows/DiscountingFxForwardEngine builds a DiscountingFxForwardEngine. A sample configuration is shown in listing 353.

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="FxForward">
    <Model>DiscountedCashflows</Model>
    <ModelParameters/>
    <Engine>DiscountingFxForwardEngine</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">FX_Analytical</Parameter>
        <Parameter name="IncludeSettlementDateFlows">true</Parameter>
    </EngineParameters>
</Product>
```

Listing 353: Configuration for Product FxForward, Model DiscountedCashflows, Engine DiscountingFxForwardEngine

CrossAssetModel/AMC builds a McCamFxForwardEngine for use in AMC simulations. We refer to the AMC module documentation for further details.

Parameter IncludeSettlementDateFlows is set to false by default, i.e. cash flows on the maturity date of the trade are not taken into account when pricing as of maturity date so that the trade has zero value on maturity. In case of true we do take the final flows into account while the trade is "maturing".

9.75 Product Type: FxOption

Used by trade type: FxOption

Available Model/Engine pairs:

- GarmanKohlhagen/AnalyticEuropeanEngine
- CrossAssetModel/AMC (for use in AMC simulations only)

Engine description:

GarmanKohlhagen/AnalyticEuropeanEngine builds a AnalyticEuropeanEngine. A sample configuration is shown in listing 354

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="FxOption">
    <Model>GarmanKohlhagen</Model>
    <ModelParameters/>
    <Engine>AnalyticEuropeanEngine</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">FX_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 354: Configuration for Product FxOption, Model GarmanKohlhagen, Engine AnalyticEuropeanEngine

CrossAssetModel/AMC builds a McCamFxOptionEngine for use in AMC simulations. We refer to the AMC module documentation for further details.

9.76 Product Type: FxOptionEuropeanCS

Used by trade type: FxOption if cash settled

Available Model/Engine pairs:
GarmanKohlhagen/AnalyticCashSettledEuropeanEngine

Engine description:

GarmanKohlhagen/AnalyticCashSettledEuropeanEngine builds a AnalyticCashSettledEuropeanEngine. A sample configuration is shown in listing 355

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="FxOptionEuropeanCS">
    <Model>GarmanKohlhagen</Model>
    <ModelParameters/>
    <Engine>AnalyticCashSettledEuropeanEngine</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">FX_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 355: Configuration for Product FxOptionEuropeanCS, Model GarmanKohlhagen, Engine AnalyticCashSettledEuropeanEngine

9.77 Product Type: FxOptionForward

Used by trade type: FxOption if physically settled and payment date is later than option expiry date

Available Model/Engine pairs: GarmanKohlhagen/AnalyticEuropeanForwardEngine

Engine description:

GarmanKohlhagen/AnalyticEuropeanForwardEngine builds a AnalyticEuropeanForwardEngine. A sample configuration is shown in listing 356

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="FxOptionForward">
    <Model>GarmanKohlhagen</Model>
    <ModelParameters/>
    <Engine>AnalyticEuropeanForwardEngine</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">FX_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 356: Configuration for Product FxOptionForward, Model GarmanKohlhagen, Engine AnalyticEuropeanForwardEngine

9.78 Product Type: FxOptionAmerican

Used by trade type: FxOption if exercise style is american

Available Model/Engine pairs:

- GarmanKohlhagen/FdGarmanKohlhagenVanillaEngine
- GarmanKohlhagen/BaroneAdesiWhaleyApproximationEngine

Engine description:

GarmanKohlhagen/FdBlackScholesVanillaEngine builds a FdBlackScholesVanillaEngine. A sample configuration is shown in listing 357

The parameters have the following meaning:

- Scheme: The finite difference scheme to use
- TimeStepsPerYear: Time grid specification
- XGrid: State grid specification
- DampingSteps: Number of damping steps taken by FD solver
- EnforceMonotoneVariance [optional]: If true variance is modified to be monotone if needed, defaults to true
- TimeGridMinimumSize [optional]: Minimum number in resulting time grid, defaults to 1 if not given
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="FxOptionAmerican">
    <Model>GarmanKohlhagen</Model>
    <ModelParameters/>
    <Engine>FdBlackScholesVanillaEngine</Engine>
    <EngineParameters>
        <Parameter name="Scheme">Douglas</Parameter>
        <Parameter name="TimeGridPerYear">100</Parameter>
        <Parameter name="XGrid">100</Parameter>
        <Parameter name="DampingSteps">0</Parameter>
        <Parameter name="EnforceMonotoneVariance">true</Parameter>
        <Parameter name="SensitivityTemplate">FX_FD</Parameter>
    </EngineParameters>
</Product>
```

Listing 357: Configuration for Product FxOptionAmerican, Model GarmanKohlhagen, Engine FdBlackScholesVanillaEngine

GarmanKohlhagen/BaroneAdesiWhaleyApproximationEngine builds a BaroneAdesiWhaleyApproximationEngine. A sample configuration is shown in listing 358

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="FxOptionAmerican">
    <Model>GarmanKohlhagen</Model>
    <ModelParameters/>
    <Engine>BaroneAdesiWhaleyApproximationEngine</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">FX_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 358: Configuration for Product FxOptionAmerican, Model GarmanKohlhagen, Engine BaroneAdesiWhaleyApproximationEngine

9.79 Product Type: FxDoubleTouchOption

Used by trade type: FxDoubleTouchOption

Available Model/Engine pairs:
GarmanKohlhagen/AnalyticDoubleBarrierBinaryEngine

Engine description:

GarmanKohlhagen/AnalyticDoubleBarrierBinaryEngine builds a AnalyticDoubleBarrierBinaryEngine. A sample configuration is shown in listing 359

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="FxDoubleTouchOption">
    <Model>GarmanKohlhagen</Model>
    <ModelParameters/>
    <Engine>AnalyticDoubleBarrierBinaryEngine</Engine>
```

```
    <EngineParameters>
        <Parameter name="SensitivityTemplate">FX_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 359: Configuration for Product FxDoubleTouchOption, Model GarmanKohlhagen, Engine AnalyticDoubleBarrierBinaryEngine

9.80 Product Type: MultiLegOption

Used by trade type: MultiLegOption

Available Model/Engine pairs:

- CrossAssetModel/MC
- CrossAssetModel/AMC

Engine description:

CrossAssetModel/MC builds a McMultiLegOptionEngine. A sample configuration is shown in listing 360

The parameters have the following meaning:

- IrCalibration: Bootstrap, BestFit, None
- IrCalibrationStrategy: CoterminalDealStrike, CoterminalATM
- ReferenceCalibrationGrid: An optional grid, only one calibration instrument per interval is kept
- IrReversion: The mean reversion (given per ccy)
- IrReversionType: Hagan, HullWhite (given per ccy)
- IrVolatility: The volatility (start value for calibration if calibrated, given per ccy)
- IrVolatilityType: Hagan, HullWhite (given per ccy)
- FxVolatility: The fx volatility (start value for calibration if calibrated, given per ccy)
- Corr_Key1_key2: The correlations for ir/fx keys (multiple entries possible)
- ShiftHorizon: Shift horizon for LGM model as fraction of deal maturity
- Tolerance: Error tolerance for calibration
- Training.Sequence: The sequence type for the traning phase, can be MersenneTwister+, MersenneTwisterAntithetc+, Sobol+, Burley2020Sobol+, SobolBrownianBridge+, Burley2020SobolBrownianBridge+
- Training.Seed: The seed for the random number generation in the training phase
- Training.Samples: The number of samples to be used for the training phase
- Pricing.Sequence: The sequence type for the pricing phase, same values allowed as for training

- Training.BasisFunction: The type of basis function system to be used for the regression analysis, can be Monomial+, Laguerre+, Hermite+, Hyperbolic+, Legendre+, Chbyshev+, Chebyshev2nd+
- BasisFunctionOrder: The order of the basis function system to be used
- Pricing.Seed: The seed for the random number generation in the pricing
- 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)
- BrownianBridgeOrdering: variate ordering for Brownian bridges, can be Steps+, Factors+, Diagonal+
- SobolDirectionIntegers: direction integers for Sobol generator, can be Unit+, Jaeckel+, SobolLevitan+, SobolLevitanLemieux+, JoeKuoD5+, JoeKuoD6+, JoeKuoD7+, Kuo+, Kuo2+, Kuo3+
- 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+
- RegressorModel: Simple, LaggedFX. If not given, it defaults to Simple. Depending on the choice the regressor is built as follows:
  - Simple: For an observation date the full model state observed on this date is included in the regressor. No past states are included though.
  - LaggedFX: For an observation date the full model state observed on this date is included in the regressor. In addition, past FX states that are relevant for future cashflows are included. For example, for a FX resettable cashflow the FX state observed on the FX reset date is included.
- SensitivityTemplate [optional]: the sensitivity template to use

---

```
<Product type="MultiLegOption">
    <Model>CrossAssetModel</Model>
    <ModelParameters>
        <Parameter name="IrCalibration">Bootstrap</Parameter>
        <Parameter name="IrCalibrationStrategy">CoterminalDealStrike</Parameter>
        <Parameter name="FxCalibration">Bootstrap</Parameter>
        <Parameter name="ReferenceCalibrationGrid">400,3M</Parameter>
        <Parameter name="IrReversion_EUR">0.0</Parameter>
        <Parameter name="IrReversionType_EUR">HullWhite</Parameter>
        <Parameter name="IrVolatility_EUR">0.01</Parameter>
        <Parameter name="IrVolatilityType_EUR">Hagan</Parameter>
        <Parameter name="IrReversion_GBP">0.0</Parameter>
        <Parameter name="IrReversionType_GBP">HullWhite</Parameter>
        <Parameter name="IrVolatility_GBP">0.01</Parameter>
        <Parameter name="IrVolatilityType_GBP">Hagan</Parameter>
        <Parameter name="FxVolatility_GBP">0.08</Parameter>
        <Parameter name="Corr_IR:EUR_FX:GBP">0.23</Parameter>
        <Parameter name="ShiftHorizon">0.5</Parameter>
        <Parameter name="Tolerance">0.20</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">0</Parameter>
        <Parameter name="BrownianBridgeOrdering">Steps</Parameter>
        <Parameter name="SobolDirectionIntegers">JoeKuoD7</Parameter>
        <Parameter name="MinObsDate">true</Parameter>
        <Parameter name="RegressorModel">Simple</Parameter>
        <Parameter name="SensitivityTemplate">IR_MC</Parameter>
    </EngineParameters>
</Product>
```

Listing 360: Configuration for Product MultiLegSwaption, Model CrossAssetModel, Engine MC

LGM/AMC builds a McMultiLegOptionEngine for use in AMC simulations. We refer to the AMC module documentation for further details.

9.81 Product Type: Swap

Used by trade type: Swap for single-currency swaps

Available Model/Engine pairs:

- DiscountedCashflows/DiscountingSwapEngine
- CrossAssetModel/AMC

Engine description:

DiscountedCashflows/DiscountingSwapEngine builds a DiscountingSwapEngine. A sample configuration is shown in listing 361

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="Swap">
    <Model>DiscountedCashflows</Model>
    <ModelParameters/>
    <Engine>DiscountingSwapEngine</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">IR_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 361: Configuration for Product Swap, Model DiscountedCashflows, Engine DiscountingSwapEngine

CrossAssetModel/AMC builds a McLgmSwapEngine for use in AMC simulations. We refer to the AMC module documentation for further details.

9.82 Product Type: CurrencySwap

Used by trade type: Swap for cross-currency swaps

Available Model/Engine pairs:

- DiscountedCashflows/DiscountingCrossCurrencySwapEngine
- CrossAssetModel/AMC

Engine description:

DiscountedCashflows/DiscountingCrossCurrencySwapEngine builds a DiscountingCurrencySwapEngine. A sample configuration is shown in listing 362

The parameters have the following meaning:

- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="CurrencySwap">
    <Model>DiscountedCashflows</Model>
    <ModelParameters/>
    <Engine>DiscountingCurrencySwapEngine</Engine>
    <EngineParameters>
        <Parameter name="SensitivityTemplate">IR_Analytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 362: Configuration for Product Swap, Model DiscountedCashflows, Engine DiscountingCurrencySwapEngine

CrossAssetModel/AMC builds a McCamCurrencySwapEngine for use in AMC simulations. We refer to the AMC module documentation for further details.

9.83 Product Type: RiskParticipationAgreement_Vanilla

Used by trade type: RiskParticipationAgreement with

- exactly one fixed and one floating leg with opposite payer flags and
- only fixed, ibor, ois (comp, avg) coupons allowed, no cap / floors, no in arrears fixings for ibor

Available Model/Engine pairs: Black/Analytic

Engine description:

Black/Analytic builds a AnalyticBlackRiskParticipationAgreementEngine. A sample configuration is shown in listing 363

The parameters have the following meaning:

- Reversion: the mean reversion parameter to use for underlying matching in a HW1F model
- MatchUnderlyingTenor: whether to match the trade underlying tenor in scenario / sensi calculations

- AlwaysRecomputeOptionRepresentation: whether to recompute underlying representation on each scenario or keep the initial representation across all scenarios
- MaxGapDays: the maximum gap between option expiry dates used for the underlying representation
- MaxDiscretizationPoints: the maximum number of discretization points used for underlying representation
- OptionExpiryPosition: optional, defaults to Mid, can be Left or Mid
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="RiskParticipationAgreement_Vanilla">
    <Model>Black</Model>
    <ModelParameters>
        <Parameter name="Reversion">0.0</Parameter>
        <Parameter name="MatchUnderlyingTenor">true</Parameter>
    </ModelParameters>
    <Engine>Analytic</Engine>
    <EngineParameters>
        <Parameter name="AlwaysRecomputeOptionRepresentation">false</Parameter>
        <Parameter name="MaxGapDays">400</Parameter>
        <Parameter name="MaxDiscretisationPoints">20</Parameter>
        <Parameter name="OptionExpiryPosition">Mid</Parameter>
        <Parameter name="SensitivityTemplate">IR_Semianalytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 363: Configuration for Product RiskParticipationAgreement_ Vanilla, Model Black, Engine Analytic

9.84 Product Type: RiskParticipationAgreement_Vanilla_XCcy

Used by trade type: RiskParticipationAgreement with

- two legs in different currencies with arbitrary coupons allowed, no optionData though (as in structured variant)

Available Model/Engine pairs: Black/Analytic

Engine description:

Black/Analytic builds a AnalyticXCcyBlackRiskParticipationAgreementEngine. A sample configuration is shown in listing 364

The parameters have the following meaning:

- AlwaysRecomputeOptionRepresentation: whether to recompute underlying representation on each scenario or keep the initial representation across all scenarios
- MaxGapDays: the maximum gap between option expiry dates used for the underlying representation

- MaxDiscretizationPoints: the maximum number of discretization points used for underlying representation
- OptionExpiryPosition: optional, defaults to Mid, can be Left or Mid
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="RiskParticipationAgreement_Vanilla_XCcy">
    <Model>Black</Model>
    <ModelParameters/>
    <Engine>Analytic</Engine>
    <EngineParameters>
        <Parameter name="AlwaysRecomputeOptionRepresentation">false</Parameter>
        <Parameter name="MaxGapDays">400</Parameter>
        <Parameter name="MaxDiscretisationPoints">20</Parameter>
        <Parameter name="OptionExpiryPosition">Mid</Parameter>
        <Parameter name="SensitivityTemplate">FX_Semianalytical</Parameter>
    </EngineParameters>
</Product>
```

Listing 364: Configuration for Product RiskParticipationAgreement_Vanilla_XCcy, Model Black, Engine Analytic

9.85 Product Type: RiskParticipationAgreement_Structured

Used by trade type: RiskParticipationAgreement with

- arbitrary number of fixed, floating, cashflow legs
- only fixed, ibor coupons, ois (comp, avg), simple cashflows allowed, but possibly capped / floored, as naked option, with in arrears fixing for ibor
- with optionData (i.e. callable underlying), as naked option (i.e. swaption)

Available Model/Engine pairs: LGM/Grid

Engine description:

LGM/Grid builds a NumericLgmRiskParticipationAgreementEngine. A sample configuration is shown in listing 365

The parameters have the following meaning:

- Calibration: Bootstrap, BestFit, None
- CalibrationStrategy: CoterminalDealStrike, CoterminalATM
- ReferenceCalibrationGrid: An optional grid, only one calibration instrument per interval is kept
- Reversion: The mean reversion
- ReversionType: Hagan, HullWhite
- Volatility: The volatility (start value for calibration if calibrated)
- VolatilityType: Hagan, HullWhite
- ShiftHorizon: Shift horizon for LGM model as fraction of deal maturity

- Tolerance: Error tolerance for calibration
- sy, sx: Number of covered standard deviations (notation as in Hagan's paper)
- ny, nx: Number of grid points for numerical integration (notation as in Hagan's paper)
- MaxGapDays: the maximum gap between option expiry dates used for the underlying representation
- MaxDiscretizationPoints: the maximum number of discretization points used for underlying representation
- OptionExpiryPosition: optional, defaults to Mid, can be Left or Mid
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="RiskParticipationAgreement_Structured">
    <Model>LGM</Model>
    <ModelParameters>
        <Parameter name="Calibration">Bootstrap</Parameter>
        <Parameter name="CalibrationStrategy">CoterminalDealStrike</Parameter>
        <Parameter name="ReferenceCalibrationGrid">400,3M</Parameter>
        <Parameter name="Reversion">0.0</Parameter>
        <Parameter name="ReversionType">HullWhite</Parameter>
        <Parameter name="Volatility">0.01</Parameter>
        <Parameter name="VolatilityType">Hagan</Parameter>
        <Parameter name="ShiftHorizon">0.5</Parameter>
        <Parameter name="Tolerance">0.02</Parameter>
    </ModelParameters>
    <Engine>Grid</Engine>
    <EngineParameters>
        <Parameter name="sy">5.0</Parameter>
        <Parameter name="ny">30</Parameter>
        <Parameter name="sx">5.0</Parameter>
        <Parameter name="nx">30</Parameter>
        <Parameter name="MaxGapDays">400</Parameter>
        <Parameter name="MaxDiscretisationPoints">20</Parameter>
        <Parameter name="OptionExpiryPosition">Mid</Parameter>
        <Parameter name="SensitivityTemplate">IR_FD</Parameter>
    </EngineParameters>
</Product>
```

Listing 365: Configuration for Product RiskParticipationAgreement_Structured, Model LGM, Engine Grid

9.86 Product Type: RiskParticipationAgreement_TLock

Used by trade type: RiskParticipationAgreement with TLock underlying.

Available Model/Engine pairs: LGM/Grid

Engine description:

LGM/Grid builds a NumericLgmRiskParticipationAgreementEngine. A sample configuration is shown in listing 366

The parameters have the following meaning:

- Calibration: Bootstrap, BestFit, None
- CalibrationStrategy: CoterminalDealStrike, CoterminalATM
- ReferenceCalibrationGrid: An optional grid, only one calibration instrument per interval is kept
- Reversion: The mean reversion
- ReversionType: Hagan, HullWhite
- Volatility: The volatility (start value for calibration if calibrated)
- VolatilityType: Hagan, HullWhite
- ShiftHorizon: Shift horizon for LGM model as fraction of deal maturity
- Tolerance: Error tolerance for calibration
- sy, sx: Number of covered standard deviations (notation as in Hagan's paper)
- ny, nx: Number of grid points for numerical integration (notation as in Hagan's paper)
- SensitivityTemplate [optional]: the sensitivity template to use

```
<Product type="RiskParticipationAgreement_TLock">
    <Model>LGM</Model>
    <ModelParameters>
        <Parameter name="Calibration">Bootstrap</Parameter>
        <Parameter name="CalibrationStrategy">CoterminalATM</Parameter>
        <Parameter name="ReferenceCalibrationGrid">400,3M</Parameter>
        <Parameter name="Reversion">0.0</Parameter>
        <Parameter name="ReversionType">HullWhite</Parameter>
        <Parameter name="Volatility">0.01</Parameter>
        <Parameter name="VolatilityType">Hagan</Parameter>
        <Parameter name="ShiftHorizon">0.5</Parameter>
        <Parameter name="Tolerance">0.02</Parameter>
        <Parameter name="CalibrationInstrumentSpacing">3M</Parameter>
    </ModelParameters>
    <Engine>Grid</Engine>
    <EngineParameters>
        <Parameter name="sy">5.0</Parameter>
        <Parameter name="ny">30</Parameter>
        <Parameter name="sx">5.0</Parameter>
        <Parameter name="nx">30</Parameter>
        <Parameter name="TimeStepsPerYear">24</Parameter>
        <Parameter name="SensitivityTemplate">IR_FD</Parameter>
    </EngineParameters>
</Product>
```

Listing 366: Configuration for Product RiskParticipationAgreement_ TLock, Model LGM, Engine Grid

9.87 Product Type: ScriptedTrade

Used by trade type: ScriptedTrade (and trades wrapped as scripted trade)

Available Model/Engine pairs: ScriptedTrade/ScriptedTrade

Engine description:

ScriptedTrade/ScriptedTrade builds a ScriptedInstrumentPricingEngine (and variants). We refer to the scripted trade module documentation for a sample configuration and more details.

9.88 Global Parameters

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

- ContinueOnCalibrationError: If set to true an exceedence of a prescribed model calibration tolerance (for e.g. the LGM model) will not cause the trade building to fail, instead a warning is logged and the trade is processed anyway. Optional, defaults to false.
- Calibrate: If false, model calibration is disabled. This flag is usually not present in a user configuration, but only used internally for certain workflows within ORE which do not require a model calibration. Optional, defaults to true.
- GenerateAdditionalResults: If false, the generation of additional results within pricing engines will be suppressed (for those pricing engines which support this). This flag is usually not present in a user configuration, but only used internally to improve the performance for processes which only rely on the NPV as a result from pricing engines, e.g. when repricing trades under sensitivity or stress scenarios. Option, defaults to false.
- RunType: Set automatically. One of NPV, SensitivityDelta, SensitivityDeltaGamma, Stress, Exposure, Capital, TradeDetails, PortfolioAnalyser, HistoricalPnL, BondSpreadImply, AbsMaturityUpdate depending on the context for which a portfolio was built. Might also be left empty. This is used by some pricing engines to adapt to certain run types. E.g. a first order sensitivity pnl expansion might be used for a SensitivityDelta run by an engine which is able to compute analytical or AAD first order sensitivities.
- McType: Set automatically. One of Classic and American depending on the context for which the pricinge engines file is referenced. Might also be left empty. This is used by some trade constructions to adapt to certain monte carlo types. E.g. Pricer would not be attached to coupons which can be computed in AMC without a leg / coupon pricer
- StrikeSpread: Defaults to 0.1. If set, overrides the default value used to calculate the strike spread for replication engines.

References

[1] BCBS-118. International Convergence of Capital Measurement and Capital Standards - A Revised Framework. *Basel Committee for Bank Supervision*, http://www.bis.org/publ/bcbs118.pdf, Nov 2005.

[2] BCBS-128. Basel Committee on Banking Supervision, *International Convergence of Capital Measurement and Capital Standards, A Revised Framework*, http://www.bis.org/publ/bcbs128.pdf, June 2006

[3] Basel Committee on Banking Supervision, *Basel III: A global regulatory framework for more resilient banks and banking systems*, http://www.bis.org/publ/bcbs189.pdf, June 2011

[4] BCBS-279. The standardised approach for measuring counterparty credit risk exposures. *Basel Committee for Bank Supervision*, http://www.bis.org/publ/bcbs279.pdf, Mar 2014.

[5] Basel Committee on Banking Supervision, *Review of the Credit Valuation Adjustment Risk Framework*, https://www.bis.org/bcbs/publ/d325.pdf, 2015

[6] Basel Committee on Banking Supervision, *Basel III: Finalising post-crisis reforms*, https://www.bis.org/bcbs/publ/d424.pdf, 2017

[7] Targeted revisions to the credit valuation adjustment risk framework. *Basel Committee for Bank Supervision*, https://www.bis.org/bcbs/publ/d507.pdf, Jul 2020.

[8] Damiano Brigo and Fabio Mercurio, *Interest Rate Models: Theory and Practice, 2nd Edition*, Springer, 2006.

[9] Michael Pykhtin, *Collateralized Credit Exposure*, in Counterparty Credit Risk, (E. Canabarro, ed.), Risk Books, 2010

[10] Michael Pykhtin and Dan Rosen, *Pricing Counterparty Risk at the Trade Level and CVA Allocations*, Finance and Economics Discussion Series, Divisions of Research & Statistics and Monetary Affairs, Federal Reserve Board, Washington, D.C., 2010

[11] Jon Gregory, *Counterparty Credit Risk and Credit Value Adjustment, 2nd Ed.*, Wiley Finance, 2013.

[12] Jon Gregory, *The xVA Challenge, 3rd Ed.*, Wiley Finance, 2015.

[13] John Gregory The xVA Challenge, Fourth Edition Wiley, 2020

[14] 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.

[15] 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

[16] Leif B. G. Andersen, Michael Pykhtin, Alexander Sokol, *Rethinking Margin Period of Risk*, http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2719964, 2016

[17] ISDA SIMM Methodology, version 2.5A, (based on v2.5a)
https://www.isda.org/a/FBLgE/ISDA-SIMM_v2.5A.pdf

[18] Andersen, L., and Piterbarg, V. (2010): Interest Rate Modeling, Volume I-III

[19] 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

[20] 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

[21] Carol Alexander, Market Risk Analysis, Volume IV, Value at Risk Models, Wiley 2009

[22] Lugannani, R.and S.Rice (1980), Saddlepoint Approximations for the Distribution of the Sum of Independent Random Variables, Advances in Applied Probability, 12,475-490.

[23] Daniels, H. E. (1987), Tail Probability Approximations, International Statistical Review, 55, 37-48.

[24] ORE Scripted Trade Module, latest version at https://github.com/OpenSourceRisk/Engine/tree/master/Docs/ScriptedTrade

[25] Andersen, L., and Buffum, D.: Calibration and Implementation of Convertible Bond Models (2002)

[26] Andersen, L., & Piterbarg, V. (2010). Interest Rate Modeling - Volume III: Products and Risk Management. Atlantic Financial Press.

[27] Andreasen J., Huge B.: Volatility Interpolation (2010)
https://ssrn.com/abstract=1694972

[28] Barone-Adesi, G., & Whaley, R. E. (1987). Efficient Analytic Approximation of American Option Values. The Journal of Finance, Vol. 42, No. 2, 301-320.

[29] Berrahoui, M. (2004). Pricing CMS spread options and digital CMS spread options with smile. Wilmott Magazine, 2004(3), 63-69.

[30] Bloomberg: OVCV Model Description (convertible bond pricing), 2017.

[31] Brigo, D., & Mercurio, F. (2006). Interest Rate Models - Theory and Practice. Berlin: Springer Finance.

[32] De Spiegeleer, J. & Schoutens, W. (2011). The Handbook of Convertible Bonds - Pricing, Strategies and Risk Management. Wiley Finance.

[33] Brockhaus, O., Long, D. (2000), *Volatility Swaps Made Simple*, RISK Technical Article

[34] Caspers, P. (2015). Farmer's CMS Spread Option Formula for Negative Rates. SSRN Electronic Journal.

[35] Caspers, Peter: Daily Spread Curves and Ester (September 30, 2019). Available at SSRN: https://ssrn.com/abstract=3500090 or http://dx.doi.org/10.2139/ssrn.3500090

[36] Castagna, M. (2006, 1 5). Consistent Pricing of FX Options. Retrieved from SSRN: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=873788

[37] Clark, I: Foreign Exchange Option Pricing, A Practitioner's Guide, Wiley Finance, 2011

[38] Demeterfi, K., Derman, E., Kamal, M., & Zou, J. (1999). A Guide to Volatility and Variance Swaps. The Journal of Derivatives 6(4), 9-32.

[39] F. Jamshidian, I. E. (2005). Replication of flexi-swaps. Journal of risk and uncertainty.

[40] Hagan, P. (2003). Convexity Conundrums: Pricing CMS Swaps, Caps and Floors. Wilmott, 38-45.

[41] Hagan, P (2004). Methodology for Callable Swaps and Bermudan "Exercise Into" Swaptions. Technical Report. http://www.researchgate.net/publication/273720052

[42] Hagan, P (200?). Evaluating and Hedging Exotic Swap Instrumnets via LGM. Article. http://www.researchgate.net/publication/265043070

[43] Haug, E. G. (1997). The Complete Guide to Option Pricing Formulas. New York: McGraw-Hill.

[44] Hull, J., & White, A. (2004), *Valuation of a CDO and nth to default CDS Without Mote Carlo Simulation*, Journal of Derivatives 12, 2

[45] Krekel, M. (2010). *Pricing distressed CDOs with Base Correlation and Stochastic Recovery*, RISK July 2010, SSRN May 2008 https://ssrn.com/abstract=11340228b

[46] Lichters, R., Stamm, R., & Gallagher, D. (2015). Modern Derivatives Pricing and Credit Exposure Analysis. Palgrave Macmillan.

[47] Merton, R. C. (1973). Theory of Rational Option Pricing. The Bell Journal of Economics and Management Science, Vol. 4, No. 1, 141-183.

[48] Open Source Risk Engine (2016-2021). opensourcerisk.org/documentation.

[49] QuantLib, a free/open-source library for quantitative finance quantlib.org

[50] Rubinstein, M., & Reiner, E. (1991). Breaking Down the Barriers. RISK Vol. 4, No. 8, 28-35.

[51] Takada, K. (2011). Valuation of Arithmetic Average of Fed Funds Rates and Construction of the US dollar Swap Yield Curve. http://ssrn.com/abstract=1981668

[52] Clark, Iain J. (2014), Commodity Option Pricing, A Practicioner's Guide, Wiley

[53] Clark, Iain J. (2011), Foreign Exchange Option Pricing, A Practicioner's Guide, Wiley

[54] Wystup, Uwe (2006), FX Options and Structured Products, Wiley

[55] Turnbull, S., & Wakeman, L. (1991). A Quick Algorithm for Pricing European Average Options. Journal of Financial and Quantitative Analysis, 26, 377-389.

[56] ISDA: Benchmark Reform and Transition from Libor https://www.isda.org/2020/05/11/benchmark-reform-and-transition-from-libor/?_zs=U36XO1&_zl=Vi0C6

[57] ISDA: User Guide to IBOR Fallbacks and RFRs http://assets.isda.org/media/3062e7b4/c133e67f-pdf/

[58] ISDA: RFR conventions and Ibor Fallbacks - Product Table http://assets.isda.org/media/4ff1a000/b6e5395e-pdf/

[59] BBG: Ibor Fallback Dashboard

[60] https://assets.bbhub.io/professional/sites/10/IBOR-Fallbacks-LIBOR-Cessation_Announcement_20210305.pdf

[61] https://www.isda.org/2021/03/05/isda-statement-on-uk-fca-libor-announcement/

[62] https://www.fca.org.uk/publication/documents/future-cessation-loss-representativeness-libor-benchmarks.pdf

[63] https://www.isda.org/2021/03/29/isda-statement-on-jbata-announcement-on-yen-tibor-and-euroyen-tibor/

[64] https://www.isda.org/a/rwNTE/CDOR-tenor-cessation_ISDA-guidance_17.11.2020_PDF.pdf

[65] Lyashenko, Andrei and Mercurio, Fabio, Looking Forward to Backward-Looking Rates: A Modeling Framework for Term Rates Replacing LIBOR (February 6, 2019). Available at SSRN: https://ssrn.com/abstract=3330240 or http://dx.doi.org/10.2139/ssrn.3330240

[66] European Equity Derivatives Research, JPMorgan Securities Ltd: Variance Swaps (17 November 2006)

[67] Dominic O'Kane (2008) Modelling Single-name and Multi-name Credit Derivatives, Wiley

Paramteter	Description
AdjustEquityForward	If false, the term $\eta h(t, e^z)$ in the coefficient of v_z in 164 is set to zero, i.e. the hazard rate h is still used in the discounting term, but the equity drift is not corrected upwards accordingly. The default value is true.
AdjustEquityVolatility	If false, the market equity volatility input is not adjusted, but directly used in the pricing model. This setting is only possible if $p = 0$. It will then set the weighting with the market survival probability $S(0, t_i)$ in the context of formula 178 to zero, i.e. V is taken as the market implied volatility without adjustment. The default value is true.
AdjustDiscounting	If false, the adjustment of the discounting rate r to the benchmark curve b is suppressed, i.e. the change of the PDE 164 to 187 is *not* made (see section "Curves used in practice" above). The default value is true.
ZeroRecoveryOverwrite	If true, the recovery rate ρ of the convertible bond is overwritten with zero. This option is usually used in conjunction with AdjustCreditSpreadToRR set to true (see below). The default value is false.
AdjustCreditSpreadToRR	If true, the credit curve $h(\cdot)$ is adjusted by a factor $\frac{1-R}{1-\rho}$ where R is the recovery rate associated to the market default curve and ρ is the recovery rate of the bond. Usually, $R = \rho$, i.e. the bond recovery rate is the same as the recovery rate of the associated credit curve. In this case, the flag has no effect, since the multiplier is 1. However, when ρ is overwritten with zero due to the flag ZeroRecoveryOverwrite set to true, the flag AdjustCreditSpreadToRR should also be set to true. The default value is false.
TreatSecuritySpreadAsCreditSpread	If true, the security spread is not incoroporated into the benchmark curve b as described above, but rather added as a spread on top of the credit curve, i.e. it is added to $h(t, e^z)$. For exchangeables, the security spread is added to *both* h^B and h^S, i.e. it simultaneously increases the credit pread of both the equity and the bond component. Since the security spread is understood as an effective discounting spread, it is scaled by $s \to s/(1-\rho)$ before it is added to h, where ρ is the recovery rate of the bond. The default value is false.

Table 41: Convertible bond model flavours