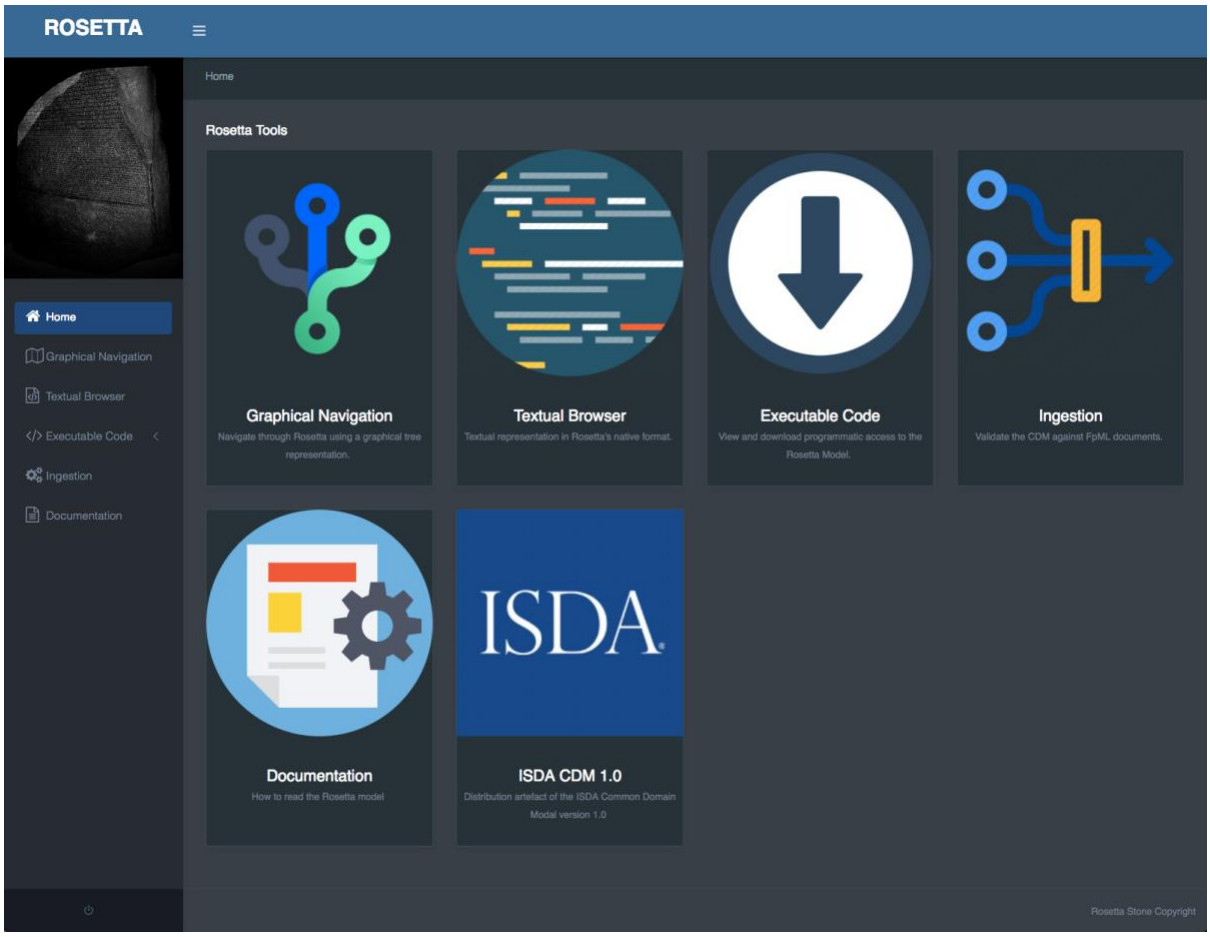
ISDA CDM 1.0 Delivery Manifest

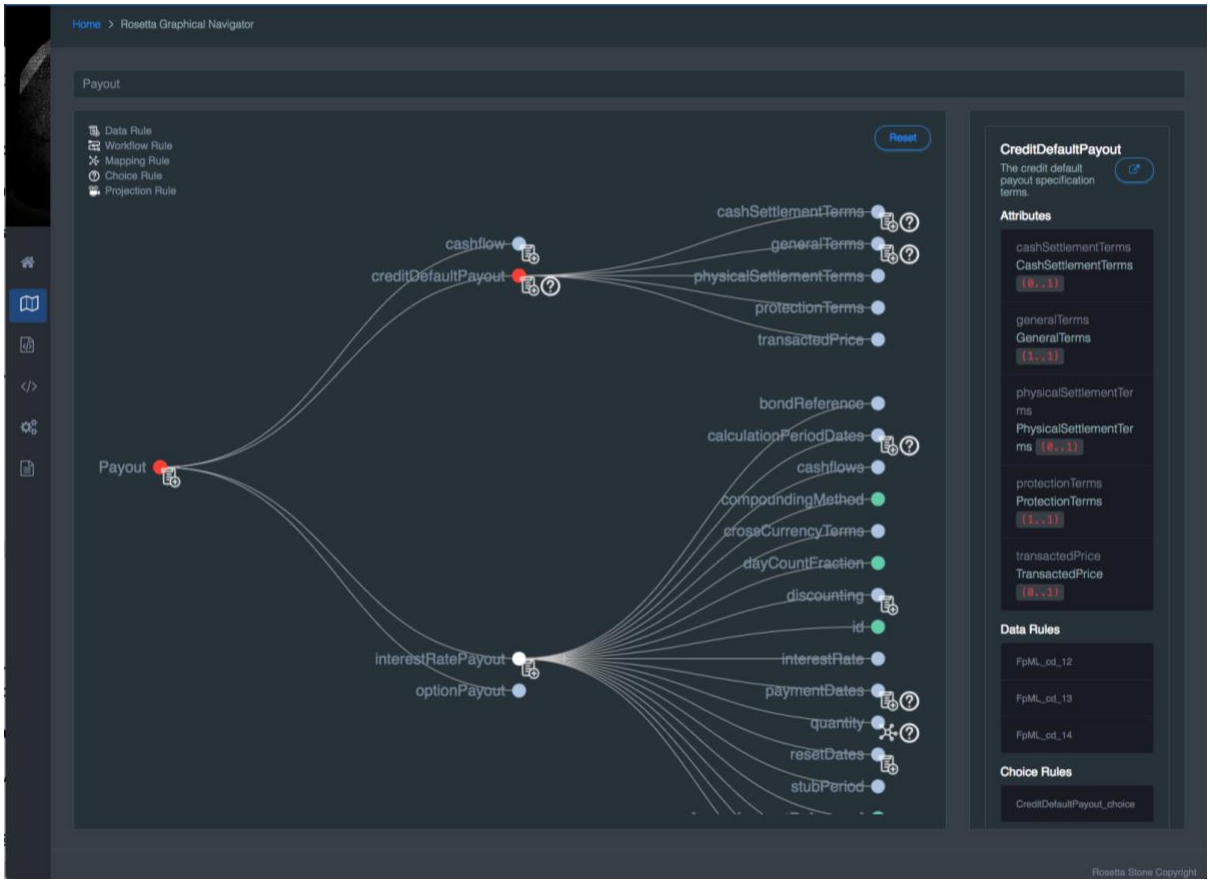
Table of Contents



Model Components Implemented as part of the Initial Phase

Products

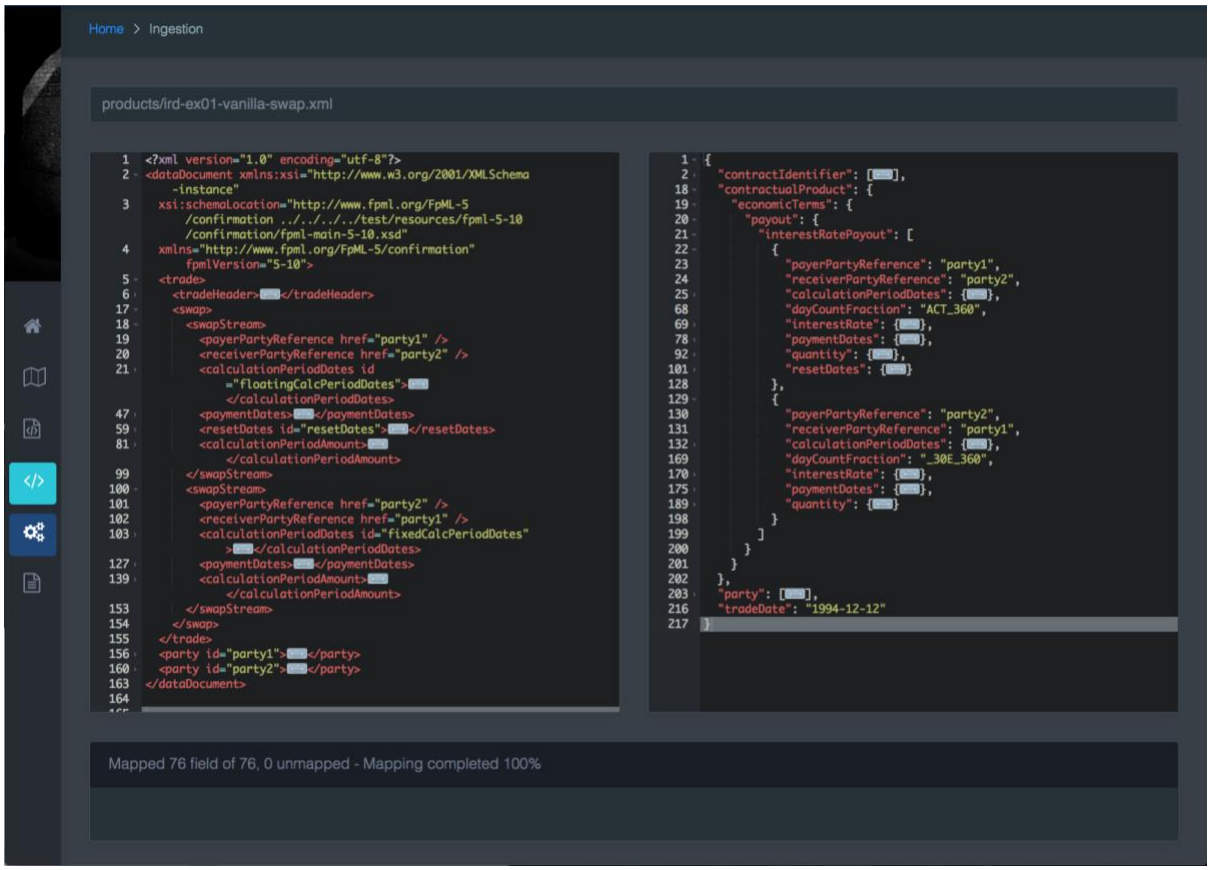
- Implemented the CDM composite paradigm which highlighted features are as follows:
    - The economic terms are specified by composition, leveraging the FpML building blocks to the extent possible while also looking for further consistency and simplicity whenever possible. As part of this initial release, 4 payout components have been specified:
        - interest rate payout;
        - credit default payout;
        - option payout;
        - cashflow.
    - The product qualification is inferred from those economic terms.
- Developed syntax to derive product type using the composite paradigm:
    - interest rate derivatives:
        - Interest rate swaps (incl. cross-currency swaps, non-deliverable swaps, basis swaps, swaps with non-regular periods, ...)
        - swaptions
        - bond and convertible bond options
    - credit derivatives
        - credit default swaps (incl. baskets, tranche, swaps with mortgage and loans underlyers, ...)
        - options on credit default swaps
- Implemented data validation logic, including a significant set of the FpML validation rules available as pseudo-code from its website.
- Mapping logic has been implemented with the latest version of the FpML standard, which has been used as part of the Rosetta Ingestion Service to automatically ingest FpML sample trades and convert them as JSON CDM instance documents.



Composite Paradigm: the Rosetta Graphical Navigator displaying Credit Default Payout and Interest Rate Payout, both represented within the Payout class. Both leveraging existing FpML components as building blocks.

```
class Payout <"The payout can be specified through a number of combinations, e.g. by associating several interest rate
    payouts to specify an interest rate swap, or a credit default and an interest rate payout to specify a credit default
    swap. The implied product is inferred by the isProduct CDM artefact.">
{
    interestRatePayout InterestRatePayout (0..*);
        [synonym FpML value swapStream pathExpression "trade.swap"]
        [synonym Rosetta_Workbench value swapStream pathExpression "contract.swap"]
        [synonym FpML value swapStream pathExpression "trade.swaption.swap"]
        [synonym Rosetta_Workbench value swapStream pathExpression "contract.swaption.swap"]
        [synonym FpML value feeLeg pathExpression "trade.creditDefaultSwap"]
        [synonym Rosetta_Workbench value feeLeg pathExpression "contract.creditDefaultSwap"]
        [synonym FpML value feeLeg pathExpression "trade.creditDefaultSwapOption.creditDefaultSwap"]
        [synonym FpML value generalTerms pathExpression "trade.creditDefaultSwap"]
        [synonym FpML value generalTerms pathExpression "trade.creditDefaultSwapOption.creditDefaultSwap"]
    creditDefaultPayout CreditDefaultPayout (0..1);
    cashflow Payment (0..*) <"A payment between the parties to the trade. For interest rate products, this corresponds to
        the FpML additionalPayment element. For credit default swaps, this corresponds to the initialPayment element and the
        singlePayment element of the fee leg.">;
        [synonym FpML value additionalPayment pathExpression "trade.swap"]
        [synonym FpML value initialPayment pathExpression "trade.creditDefaultSwap.feeLeg"]
        [synonym FpML value singlePayment pathExpression "trade.creditDefaultSwap.feeLeg"]
        [synonym FpML value premium pathExpression "trade.swaption"]
        [synonym Rosetta_Workbench value premium pathExpression "contract.swaption"]
        [synonym FpML value premium pathExpression "trade.creditDefaultSwapOption"]
        [synonym FpML value premium pathExpression "trade.bondOption"]
    optionPayout OptionPayout (0..*);
}
```

Rosetta Textual Browser displaying the definition of the Payout class, which is comprised of the various payout representations. Additionally, in the view are Synonym definitions which map attributes from the FpML standard and a customised event schema to the CDM canonical model.



The Rosetta Ingestion UI showing a side-by-side comparison between a vanilla interest Rate Swap FpML document (from the FpML.org website) on the left and the corresponding CDM JSON document on the right. Note that the product representation in CDM no longer qualifies an interest rate swap by name; rather, this is inferred from the existence of two Interest Rate Payout classes within the Economic Terms.

```
isProduct InterestRate_IRSwap_Basis
    [synonym ISDA_Taxonomy_v1 value InterestRate_IRSwap_Basis]
    [synonym Bank_A value BasisSwap] // The purpose is to illustrate how firms could map to their own internal product
        identification
    [synonym Venue_B value FloatFloatInterestSwap]
    EconomicTerms -> payout -> interestRatePayout -> interestRate -> floatingRate exists
    and EconomicTerms -> payout -> interestRatePayout -> interestRate -> floatingRate exists
```

The qualification of a Fixed-Float Interest Rate Swap: the Economic Terms of a contract contain one Interest Rate Payout with a Fixed Rate and another Interest Rate Payout with a Floating Rate.

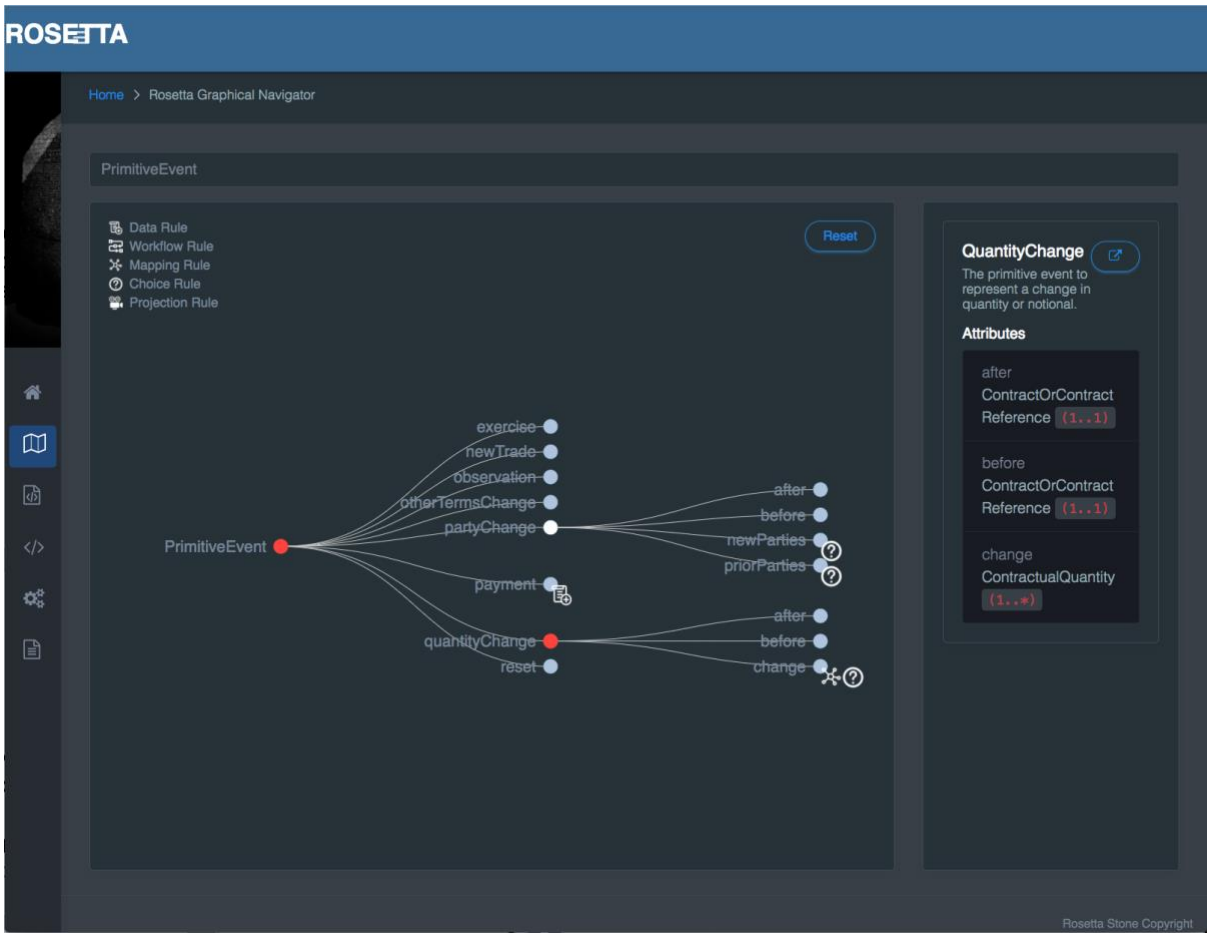
Sample products ingested as part of the Rosetta Workbench

Asset Class	Highlight	Source	File Name
Credit	Single name credit default swap with reference obligation, periodic payment schedule, physical settlement terms and referencing ISDA 1999 definitions	FpML sample	cd-ex01-long-asia-corp-fixreg
Credit	Single name credit default swap with reference obligation, periodic payment schedule and referencing ISDA 2003 definitions	FpML sample	cd-ex02-2003-short-asia-corp-fixreg
Credit	Single name credit default swap with reference obligation, periodic payment, cash settlement terms schedule and referencing ISDA 1999 definitions	FpML sample	cd-ex16-short-us-corp-fixreg-recovery-factor
Credit	Index credit default swap with periodic payment schedule and initial payment	FpML sample	cdindex-ex01-cdx
Credit	Index credit default swap with initial payment but no periodic payment schedule	FpML sample	cdindex-ex02-iTraxx
Credit	Index credit default swap with initial payment, no periodic payment schedule, master agreement documentation and account information	FpML sample	cdindex-ex03-iTraxx-contractual-supplement
Credit	Single name credit default swap with reference obligation, periodic payment schedule, physical settlement terms, collateral terms and referencing ISDA 2003 North American master confirmation template	FpML sample	cd-indamt-ex01-short-us-corp-fixreg
Credit	Single name credit default swap with revolving loan reference, periodic payment schedule and collateral terms	FpML sample	cds-ELCDS-ReferenceObligation
Credit	European-style put swaption on a single name credit default swap, with knockout feature and strike price expressed in reference to the fixed rate of the underlying swap	FpML sample	cd-swaption-1
Credit	European-style put swaption on a single name credit default swap, with knockout feature and strike price expressed in reference to the fixed rate of the underlying swap	FpML sample	cd-swaption-2
Credit	European-style credit default swap option with the strike expressed as a spread	FpML sample	itraxx-index-option
Credit	Mortgage credit default swap, with physical settlement terms, single payment and referencing the 2003 credit definitions	FpML sample	cdm-cds-mortgage-CMBS-single-payment
Credit	Single name credit default swap with reference obligation and periodic payment schedule	GS sample	cdm-cds-ref-ob
Credit	Basket credit default swap with reference obligations, tranche specification, periodic payment and reference to the ISDA 2003 definitions	FpML sample	cds-basket-tranche
Credit	Basket credit default swap with reference obligations, periodic payment and reference to the ISDA 2003 definitions	FpML sample	cds-basket
Credit	Single name credit default swap with revolving loan reference, periodic payment schedule and physical settlement terms	FpML sample	cds-loan-ReferenceObligation
Credit	Single name credit default swap with no specified reference obligation but reference to the	FpML sample	cds-loan-SecuredList

	Secured List as specified in the 2003 ISDA Definitions		
Credit	Residential mortgage credit default swap with periodic payment schedule, physical settlement terms and master document reference	FpML sample	cds-mortgage-RMBS
Credit	European-style credit default swap option on an index swap with the strike expressed as a spread and a reference to the ISDA 2003 definitions	FpML sample	cdx-index-option
Interest Rates	Fixed/Float vanilla IRS	FpML sample	ird-ex01-vanilla-swap
Interest Rates	Fixed/float IRS with amortizing notional schedule, initial stub with implied rate interpolation and cashflow representation of the swap stream	FpML sample	ird-ex02-stub-amort-swap
Interest Rates	Fixed/float IRS with flat compounding terms and cashflow representation of the swap stream	FpML sample	ird-ex03-compound-swap
Interest Rates	Fixed/float IRS with fixed rate schedule, set in arrear reset date and initial fee	FpML sample	ird-ex04-arrears-stepup-fee-swap
Interest Rates	Fixed/float IRS with final stub	FpML sample	ird-ex05-long-stub-swap
Interest Rates	Fixed/float IRS with EUR-EONIA-OIS-COMPOUND floating rate reference and Term as the calculation and payment period reference	FpML sample	ird-ex07-ois-swap
Interest Rates	European-style swaption with manual exercise procedure and underlying Fixed/Float IRS	FpML sample	ird-ex09-euro-swaption-explicit
Interest Rates	European-style swaption straddle with cash settlement terms, incl. the parYieldCurveUnadjustedMethod, and with an underlying Fixed/Float IRS	FpML sample	ird-ex12-euro-swaption-straddle-cash
Interest Rates	Fixed/float cross-currency IRS with FX-linked notional schedule	FpML sample	ird-ex25-fxnotional-swap
Interest Rates	Fixed/float non-deliverable IRS with a MoneyMarketYield rate treatment provision	FpML sample	ird-ex29-non-deliverable-settlement-swap
Interest Rates	Fixed/float IRS with relative effective date and termination dates, compounding provision, customised cashflow schedule and weighted floating rate observation schedule	FpML sample	ird-ex30-swap-comp-avg-relative-date
Interest Rates	Fixed/float IRS referencing the BRL-CDI floating rate index, with flat compounding method, a non-deliverable settlement clause and a future value notional	FpML sample	ird-ex33-BRL-CDI-swap
Interest Rates	CAD basis swap 3M/1M with spread and initial stub	LCH sample	CAD Long Initial Stub
Interest Rates	Fixed/float cross-currency IRS with FX-linked notional schedule	GS sample	cdm-xccy-swap-after
Interest Rates	Fixed/float cross-currency IRS with FX-linked notional schedule	GS sample	cdm-xccy-swap-before
Interest Rates	European-style bond option	FpML sample	bond-option
Interest Rates	American-style convertible bond option	FpML sample	cb-option

Events

- Implemented the CDM composite paradigm alongside the same principles as the products.
- Developed syntax to use the composite event model to derive the event qualification for the following set of events:
    - Allocation
    - Compression
    - Exercise
    - Full termination
    - New trade
    - Novation
    - Observation
    - Partial termination
    - Payment
    - Reset
- Specified an XML event representation as a way to test the ingestion of external events along the same principles as those developed for the products and ingested a set of manually crafted events as part of the Ingestion Service.



Composite Paradigm: the Event Model is based on the definition of Primitive Events, which are used with 'isEvent' logic to qualify more complex events such as Partial Termination and Novation.

```
class QuantityChange <"The primitive event to represent a change in quantity or notional.">
{
    before ContractOrContractReference (1..1);
        [synonym Rosetta_Workbench value before]
    after ContractOrContractReference (1..1);
        [synonym Rosetta_Workbench value after]
    change ContractualQuantity (1..*);
        [synonym Rosetta_Workbench value change]
}
```

Representation of a Quantity Change Event, which is used to qualify a Partial Termination Event.

```
isEvent PartialTermination <"The qualification of a partial termination event.">
    Event -> primitive -> quantityChange exists
    and Event -> intent = IntentEnum.partialTermination
    and NotionalAmount_Decrease, NotionalAmount_Remaining apply
```

```
data rule NotionalAmount_Decrease <"Logic to qualify a decrease in the notional amount as a result of a quantity change
    primitive event.">
    when Event -> primitive -> quantityChange -> before -> contract -> contractualProduct -> economicTerms -> payout ->
        interestRatePayout -> quantity -> notionalAmount exists
    then Event -> primitive -> quantityChange -> before -> contract -> contractualProduct -> economicTerms -> payout ->
        interestRatePayout -> quantity -> notionalAmount -> amount >
        Event -> primitive -> quantityChange -> after -> contract -> contractualProduct -> economicTerms -> payout ->
            interestRatePayout -> quantity -> notionalAmount -> amount
```

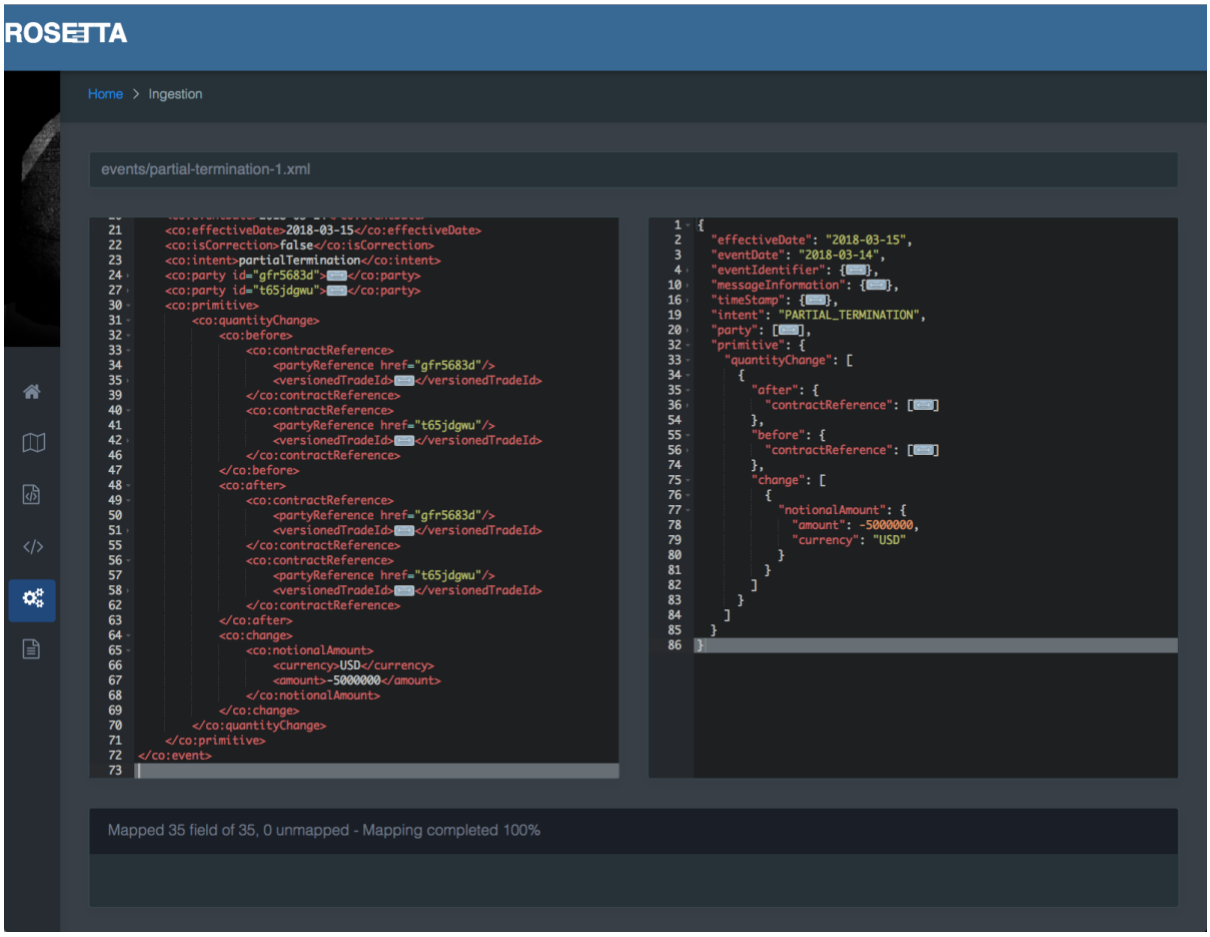
```
data rule NotionalAmount_Remaining <"Logic to qualify a remaining notional amount as a result of a quantity change primitive
    event.">
    when Event -> primitive -> quantityChange -> after -> contract -> contractualProduct -> economicTerms -> payout ->
        interestRatePayout -> quantity -> notionalAmount exists
    then Event -> primitive -> quantityChange -> after -> contract -> contractualProduct -> economicTerms -> payout ->
        interestRatePayout -> quantity -> notionalAmount -> amount > 0
```

Derived Event Types: using CDM syntax, a Partial Termination event is defined as a Quantity Change event where the quantity decreased and the amount remaining is greater than zero. NB: the syntax can be extended to represent additional quantity representations.

Sample events ingested as part of the Rosetta Workbench

Features	Source	File Name
Allocation event	REGnosys	allocation
Bilateral compression event	REGnosys	compression-bilateral
Cash exercise of a swaption	REGnosys, with GS provided trade	exercise-swaption-cash
Physical exercise of a swaption	REGnosys	exercise-swaption-physical
Full termination event, with a reference to the 'after' contract	REGnosys	full-termination-contract-reference
Full termination event, with an explicitly stated 'after' contract	REGnosys	full-termination-contract
New trade event	REGnosys	new-trade-1
Novation event involve the 2 parties	REGnosys	novation-both-parties
Payment event associated with the 2 parties novation event	REGnosys	novation-both-parties-payment-1
Payment event associated with the 2 parties novation event	REGnosys	novation-both-parties-payment-2
Full novation event with 1 party change	REGnosys	novation -full
Partial novation event with 1 party change	REGnosys	novation-partial
Payment event associated with the full novation event with 1 party change	REGnosys	novation-payment-full
Payment event associated with the partial novation event with 1 party change	REGnosys	novation-payment-partial

Observation event	REGnosys	observation-1
Observation event making use of multiple observations and with an associated interpolation function call	REGnosys	observation-derived
Partial termination event with no associated cashflow	REGnosys	partial-termination-reference-contract-no-cash
Partial termination event with associated cashflow	REGnosys	partial-termination-contract-cash
Partial termination event of a cross-currency swap with 2 associated cashflows	REGnosys, with GS provided trade	partial-termination-xccy
Payment event	REGnosys	payment-1
Reset event	REGnosys	reset-1
Bundle event that combines a new trade, an observation event and a reset event, for the purpose of testing the interest calculation logic as part of the Rosetta Ingestion Service	REGnosys	reset-bundle



Validation and Completeness: the Rosetta Ingestion UI takes a sample XML document and creates the CDM JSON representation in real-time. The status summary at the bottom confirms all 35 fields from the XML document were ingested into a CDM JSON document.

Interest calculation

- Implemented the ISDA 2006 definitions for the Fixed Amount and the Floating Amount Definitions using the CDM syntax and model components, and generated executable Java code.
- Calculations are unambiguous, machine readable and executable.
- Made use of this generated code as part of the Ingestion Service to ingest bundled events that combine trade, observation and reset artefacts to compute the accrued interest.

```
calculation FixedAmount <"2006 ISDA Definition Article 4 Section 4.4 'Fixed Amount' means, in respect of a Swap Transaction
    and a Fixed Rate Payer, an amount that, subject to any other applicable provisions, is payable by that Fixed Rate Payer
    on an applicable Payment Date and is specified in the related Confirmation or is determined as provided in Article 5 of
    these 2006 Definitions or as provided in the related Confirmation.">
{
    fixedAmount number: calculationAmount * rate * dayCountFraction
    currencyAmount CurrencyEnum: currencyAmount
}

arguments FixedAmount <"2006 ISDA Definition Article 4 Section 4.8. Calculation Amount. 'Calculation Amount' means, in
    respect of a Swap Transaction and a party, the applicable Notional Amount or Currency Amount, as the case may be. Section
    4.6. 'Currency Amount' means, in respect of a party and any Calculation Period for a Swap Transaction involving more than
    one currency, the amount specified as such for the Swap Transaction or that party.">
{
    calculationAmount: is InterestRatePayout -> quantity -> notionalSchedule -> notionalStepSchedule -> initialValue
    currencyAmount: is InterestRatePayout -> quantity -> notionalSchedule -> notionalStepSchedule -> currency
    rate: is InterestRatePayout -> interestRate -> fixedRate -> initialValue
    dayCountFraction: is InterestRatePayout -> dayCountFraction
}
```

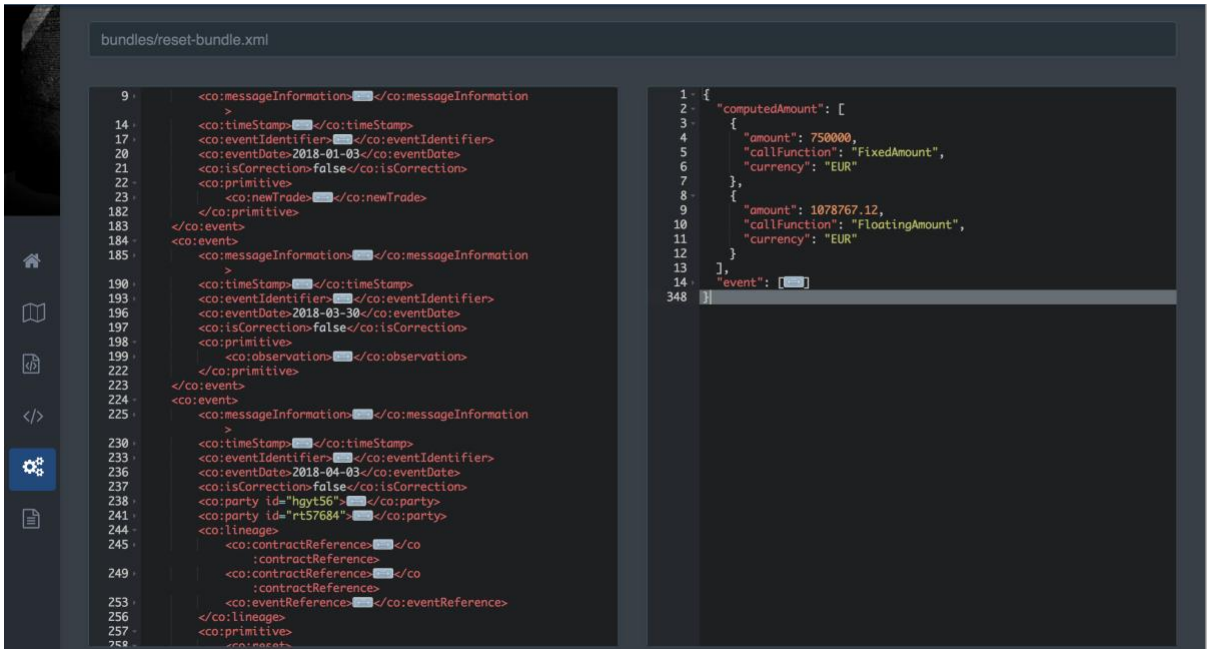
Machine Readable and Executable Definitions: (top) The Fixed Amount Calculation definition and (below) the Fixed Amount Calculation Arguments. Each argument is expressed as a path from a model object to the location of the corresponding argument value.

```
calculation DayCountFractionEnum._30E_360 <"2006 ISDA Definition Article 4 section 4.16(g): If '30E/360' or 'Eurobond Basis'
    is specified, the number of days in the Calculation Period or Compounding Period in respect of which payment is being
    made divided by 360, calculated on a formula basis as follows:[[360 x (Y2 - Y1)] + [30 x (M2 - M1)] + (D2 - D1)]/360">
{
    number: (360 * (endYear - startYear) + 30 * (endMonth - startMonth) + (endDay - startDay)) / 360
}

arguments DayCountFractionEnum._30E_360 <"2006 ISDA Definition Article 4 section 4.16(g). 'Y1' is the year, expressed as a
    number, in which the first day of the Calculation Period or Compounding Period falls; 'Y2' is the year, expressed as a
    number, in which the day immediately following the last day included in the Calculation Period or Compounding Period
    falls; 'M1' is the calendar month, expressed as a number, in which the first day of the Calculation Period or Compounding
    Period falls; 'M2' is the calendar month, expressed as a number, in which the day immediately following the last day
    included in the Calculation Period or Compounding Period falls; 'D1' is the first calendar day, expressed as a number, of
    the Calculation Period or Compounding Period, unless such number would be 31, in which case D1 will be 30; and 'D2' is
    the calendar day, expressed as a number, immediately following the last day included in the Calculation Period or
    Compounding Period, unless such number would be 31, in which case D2 will be 30.">
{
    alias period CalculationPeriod( InterestRatePayout -> calculationPeriodDates )

    endYear : is period -> endDate -> year
    startYear : is period -> startDate -> year
    endMonth : is period -> endDate -> month
    startMonth : is period -> startDate -> month
    startDay : is Min( period -> startDate -> day, 30 )
    endDay : is Min( period -> endDate -> day, 30 )
}
```

The 30E/360 Day Count Fraction calculation according to ISDA 2006 Definitions. Used within the calculation chain to evaluate the Fixed Amount Calculation.



Executable Code: the auto-generated Java code for the Fixed Amount and Floating Amount Calculations are executed by the Rosetta Ingestion Service using the sample XML document (left) and the result (right) displayed in the Rosetta Ingestion UI.

Packaged Artefacts

The packaged artefacts associated with CDM 1.0 is available for download for those who have been entitled with access to the Rosetta portal, and is composed of:

1. CDM model files in .rosetta format;
2. Generated Java classes, packaged within a Java Archive in .jar format;
3. Source code for generated Java classes, packaged within a Java archive in .jar format;
4. Upstream dependencies of the generated Java classes in .jar format;
5. Updated documentation in .html format.

Given the Java classes, a member can start to create CDM products and events for evaluation and integration.

This provides on-going access to this CDM 1.0 version, which will not be affected by the further model enhancements. (The ability to get similar artefacts available from the latest CDM beta release will be evaluated, as a way to access to such enhancements.)

In the Git software repository where the software is currently hosted, the code will be tagged with reference to this version 1.0.

As a further step, we could look to move the Git CDM repository within the ISDA organisation as a way to provide proper oversight onto it.