CreditAnalytics Feature Spec Guide

Lakshmi Krishnamurthy

v1.5 – 22 May 2012

Package org.drip.analytics.core

Serializer

1. This interface defines the core object serializer methods – serialization into and de-serialization out of byte arrays.
2. Methods that the serializer implements provide the serializer version, as well as object trailers, and delimiters for fields, collection records, collection key values, and multi-level key-value collections. Derived implementations over-ride these fields as appropriate.

Package org.drip.analytics.curve

CreditCurve

1. Created the baseline hazard curve holder object – called CreditCurve. Also contains term structure for recovery, the calibration instruments, calibration measures, calibration quotes, and parameters.
2. Create survival/recovery curve from start date, a solitary hazard rate and a solitary recovery.
3. Create survival/recovery curve from start date, survival nodes, and a solitary recovery.
4. Credit Curve from start date, single hazard node and solitary recovery.
5. Create credit curve from start date, hazard node array, and solitary recovery
6. Create credit curve from start date, hazard node array and recovery node array. Node points need not match for hazards and recoveries.
7. Set credit curve calibration instruments, calibration valuation parameters, calibration pricing parameters, calibration market parameters (discount curves/fixings), calibration measure, and calibration type.
8. Retrieve the quote for a given input calibration instrument.
9. Calibrate and create a new parallel hazard-shifted curve.
10. Calibrate and create a new parallel quote-shifted curve.
11. Calibrate and create a new parallel SNAC quote shifted curve.
12. Get the credit curve name
13. Set calibration node quote value.
14. Bump the calibration node quote
15. Assign a flat hazard node value across all tenors.
16. Calculate survival from the start date to the given date/tenor
17. Calculate time weighted survival from the start date to dates within a pair of start and end dates/tenors

18. Calculate the hazard rate between two dates/tenors
19. Calculate hazard rate to the specified date.
20. Calculate recovery rate at the specified date/tenor.
21. Calculate time weighted recovery rate from the start date to dates within a pair of start and end dates/tenors
22. Retrieve the calibration instruments
23. Get the quotes corresponding to the calibration instruments.
24. Pull the latest corresponding quotes for the calibration instruments.
25. Implements the serialization interface to serialize the credit curve instance into byte arrays, as well as de-serialize and populate a credit curve instance from an input byte array.

DiscountCurve

1. Created the baseline discount curve holder object – called DiscountCurve. Also contains term structure for the forward rates, the calibration instruments, calibration measures, calibration quotes, and parameters.
2. Build the discount curve from start date, currency, and an array of node of discount factors.
3. Build the discount curve from start date, currency, and a discount rate.
4. Build the discount curve from start date, currency, and an array of dates and discount rates.
5. Set discount curve calibration instruments, calibration valuation parameters, calibration pricing parameters, calibration market parameters (fixings), calibration measure, and calibration type.
6. Retrieve the quote for a given input calibration instrument.
7. Calibrate and create a new parallel quote-shifted curve.
8. Calibrate and create a new parallel rate-shifted curve.
9. Calibrate a shifted curve using a maturity-mismatched basis.
10. Get the discount curve name.

11. Get the discount curve currency.
12. Get the discount factor to a given date/tenor.
13. Calculate time weighted discount factor from the start date to dates within a pair of start and end dates/tenors.
14. Set calibration node quote value.
15. Bump the calibration node quote
16. Assign a flat hazard node value across all tenors.
17. Imply a rate between two dates/tenors.
18. Retrieve the calibration instruments
19. Get the quotes corresponding to the calibration instruments.
20. Pull the latest corresponding quotes for the calibration instruments.
21. Implements the serialization interface to serialize the discount curve instance into byte arrays, as well as de-serialize and populate a discount curve instance from an input byte array.

FXBasis

1. FXBasis curve representing term structure of FX basis. Basis can be full or boot-strapped.
2. Constructor from currency pair, spot date/FX, dates/basis array, flag indicating whether the basis has been boot-strapped.
3. Retrieves the currency pair
4. Get the spot date
5. Get the spot FX
6. Indicates whether the FX basis is boot-strapped
7. Calculates the term structure of the FX forward (as either outright or as PIP) from a pair of domestic/foreign discount curves, from either domestic or foreign basis.
8. Implements the serialization interface to serialize the FXBasis curve instance into byte arrays, as well as de-serialize and populate a FXBasis curve instance from an input byte array.

FXCurve

1. Contains the term structure of dates/times and FX forwards (pips/outright), and Spot FX info for the given currency pair.
2. FX Curve constructor from date/time/FX fwd/isPIP term structure, currency pair, and spot FX info.
3. Retrieve the currency pair
4. Retrieve the spot date
5. Retrieve the Spot FX.
6. Calculate the full basis across the entire term of the forward curve give the domestic and foreign rates curves, for the input valuation parameters.
7. Boot-strap the constant forward basis across the entire term of the forward curve given the domestic and foreign rates curves, for the input valuation parameters.
8. Boot-strap a new domestic or foreign discount curve using the entire term of the forward curve given the domestic and foreign rates curves, for the input valuation parameters.
9. Implements the serialization interface to serialize the FXCurve instance into byte arrays, as well as de-serialize and populate a FXCurve instance from an input byte array.

ZeroCurve

1. ZeroCurve contains the baseline zero discount curve holder object. It is primarily used for calibrating the Z Spread of a bond. It maintains term structure for the spot zero rates, the current discount factors, the accrual fractions, and the corresponding dates.

2. ZeroCurve is constructed from the bond's cash flow list, the exercise parameters, the reference discount curve, the yield quoting convention, and the optional flat bump to the discount rate.
3. ZeroCurve overrides the getDF function of the DiscountCurve, and provides the ability to get the zero rate at one of the given pre-set nodes.

Package org.drip.analytics.daycount

ActActDCParams

1. Contains the date parameters corresponding to the actual/actual reference period.
2. Implements the serialization interface to serialize the ActActDCParams object instance into byte arrays, as well as de-serialize and populate a ActActDCParams instance from an input byte array.

DateAdjustParams

1. Contains the parameters needed for adjusting dates – holiday calendar and adjustment type.
2. Implements the serialization interface to serialize the DateAdjustParams object instance into byte arrays, as well as de-serialize and populate a DateAdjustParams instance from an input byte array.

DayCountBasis

1. Contains flags that indicate where the holidays are loaded from, as well as the holiday types and load rules.
2. Loads the holiday calendar from the specified location.
3. Get the available holiday locations
4. Get the weekend days corresponding to a calendar set.
5. Get the week days corresponding to a calendar set.
6. Get the available day count conventions.
7. Calculate the year fraction between 2 days for the given day count.

8. Roll the given date according to the date roll convention and the holiday calendar set.
9. Check if the given date is a holiday according to the holiday calendar set.
10. Calculate the number of business days between two days according to the holiday calendar set.
11. Calculate the number of holidays between two days according to the holiday calendar set.
12. Adjust the given date forward in accordance with the given holiday calendar set.

FixedHoliday

1. Contains the fixed holiday's date and month. Will be generated with an optional adjustment for weekends in a given year.
2. Implements the serialization interface to serialize the FixedHoliday object instance into byte arrays, as well as de-serialize and populate a FixedHoliday instance from an input byte array.

FloatingHoliday

1. Contains the floating holiday's month, day in week, and week in month. Will be generated with an optional adjustment for weekends in a given year.
2. Implements the serialization interface to serialize the FloatingHoliday object instance into byte arrays, as well as de-serialize and populate a FloatingHoliday instance from an input byte array.

Holiday

1. Abstraction around holiday and description. Abstract function generates an optional adjustment for weekends in a given year.

2. Rolls around a weekend – rolling to a preceding or succeeding day depending upon whether it is a first or second weekend day.
3. Get the holiday description.
4. Implements the serialization interface to serialize the Holiday object instance into byte arrays, as well as de-serialize and populate a Holiday instance from an input byte array.

LocHolidays

1. Contains the set of regular holidays and the weekend holidays for a location.
2. Set the weekend as either an array of days, or as the standard weekend (Saturday and Sunday).
3. Adds static, fixed, or floating holidays.
4. Retrieves the weekend and the regular holiday set.

StaticHoliday

1. Contains a full date as a fixed holiday. Can be constructed from a stringified date or a Julian date.
2. Implements the serialization interface to serialize the StaticHoliday object instance into byte arrays, as well as de-serialize and populate a StaticHoliday instance from an input byte array.

WeekendHoliday

1. Contains the dates corresponding to the weekend. Can be set from a date array, or as standard (Saturday and Sunday).
2. Retrieves the weekend days.

3. Identifies if the given date is a left or a right weekend.
4. Identifies if the date is a weekend.
5. Implements the serialization interface to serialize the WeekendHoliday object instance into byte arrays, as well as de-serialize and populate a WeekendHoliday instance from an input byte array.

Package org.drip.analytics.period

CouponPeriod

1. Extends the period class with a few day-count specific parameters such as: frequency, reset date, and accrual day-count convention.
2. Gets the accrual fraction to an arbitrary date within the period, as well the period reset date.
3. Helper function to construct a single coupon period from start and end dates (as in a zero coupon bond).
4. Helper function to create a set of coupon periods according generated either backwards or forwards, according to period frequency, and optionally specific date adjustment rules for every date set (start/end dates, accrual start/end dates, pay date, and reset date). Also allows for accrual DCF different from coupon DCF.
5. Implements the serialization interface to serialize the CouponPeriod object instance into byte arrays, as well as de-serialize and populate a CouponPeriod instance from an input byte array.

Period

1. Holder for the period dates: period start/end, period accrual start/end, pay, and full period day count fraction.
2. Retrieve the period start/end, period accrual start/end, pay, and full period day count fraction. Period reset date defaults to the period start date.
3. Gets the accrual fraction to an arbitrary date within the period.
4. Implements the serialization interface to serialize the Period object instance into byte arrays, as well as de-serialize and populate a Period instance from an input byte array.

ProductCouponPeriodCurveMeasures

1. Period class enhanced by the following period measures: start/end survival probabilities, start/end notionals, and period end discount factor.
2. Retrieves period's start/end survival probabilities, start/end notionals, and period end discount factor.
3. Implements the serialization interface to serialize the ProductCouponPeriodCurveMeasures object instance into byte arrays, as well as de-serialize and populate a ProductCouponPeriodCurveMeasures instance from an input byte array.

ProductLossPeriodCurveMeasures

1. Period class enhanced by the following period measures: start/end survival probabilities, period effective notional/recovery/discount factor.
2. Retrieves period's start/end survival probabilities, period effective notional/recovery/discount factor.
3. Implements the serialization interface to serialize the ProductLossPeriodCurveMeasures object instance into byte arrays, as well as de-serialize and populate a ProductLossPeriodCurveMeasures instance from an input byte array.

Package org.drip.calc.output

BasketOutput

1. Placeholder for analytical basket measures, optionally across scenarios. Contains measure maps for unadjusted base IR/credit curves, flat delta/gamma bump measure maps for IR/credit bump curves, component/tenor bump double maps for IR/credit curves, flat/component recovery bumped measure maps for recovery bumped credit curves.
2. Implements the serialization interface to serialize the BasketOutput object instance into byte arrays, as well as de-serialize and populate a BasketOutput instance from an input byte array.

BondCouponMeasures

1. This class encapsulates the parsimonious, but complete set of the cash-flow oriented coupon measures generated out of a full run to a set of given work-out parameters.
2. Specifically, it is a placeholder for the “dirty” measures – dirty DV01, dirty coupon PV, and full dirty PV.
3. Adjusts for settlement, and for clean/dirty accrual.
4. Uploads the state in the fields onto a named measure map.
5. Implements the serialization interface to serialize the BondCouponMeasures object instance into byte arrays, as well as de-serialize and populate a BondCouponMeasures instance from an input byte array.

BondOutput

1. Placeholder for a comprehensive suite of analytical bond measures. Contains bid/ask sides for price, yield, G spread, Z spread, I spread, spread to treasury benchmark, par asset swap spread, workout dates, workout factors, and credit basis.
2. Can re-construct the object from a string representation, as well as stringify it.
3. Implements the serialization interface to serialize the BondOutput object instance into byte arrays, as well as de-serialize and populate a BondOutput instance from an input byte array.

BondRVMeasures

1. This class encapsulates the comprehensive set of RV measures generated out of a full run to a set of given work-out parameters.
2. Holds the price, the Z Spread, the I Spread, the G Spread, the Spread to treasury benchmark, the bond basis, the par asset swap spread, the credit basis, the duration, the convexity, and the work-out information.
3. The constructor automatically sets the full state.
4. Uploads the state in the fields onto a named measure map.
5. Implements the serialization interface to serialize the BondRVMeasures object instance into byte arrays, as well as de-serialize and populate a BondRVMeasures instance from an input byte array.

BondWorkoutMeasures

1. This class encapsulates the comprehensive set of scenario measures generated out of a full run to a set of given work-out parameters.
2. Holds the clean/dirty credit risky and risk-less bond coupon measures, risky and risk-less par and principal PV, recovery PV, expected recovery, default exposure with and without recovery, loss on instantaneous default, dirty accrued 01, first coupon rate, and first index rate.

3. The constructor automatically sets the full state.
4. Uploads the state in the fields onto a named measure map.
5. Implements the serialization interface to serialize the BondWorkoutMeasures object instance into byte arrays, as well as de-serialize and populate a BondWorkoutMeasures instance from an input byte array.

ComponentOutput

1. Placeholder for analytical single component output measures, optionally across scenarios. Contains measure maps for unadjusted base IR/credit curves, flat delta/gamma bump measure maps for IR/credit bump curves, tenor bump double maps for IR/credit curves, flat/recovery bumped measure maps for recovery bumped credit curves.
2. Implements the serialization interface to serialize the ComponentOutput object instance into byte arrays, as well as de-serialize and populate a ComponentOutput instance from an input byte array.

Package org.drip.chart.surface

BuildSurface

1. Construct an OpenGL 3D surface chart for the z's corresponding to a set of (x, y).

Contour3D

1. Construct an OpenGL 3D mapped contour surface chart for the z's corresponding to a set of (x, y).

ContourPlots

1. Construct an ortho-normalized OpenGL 3D color mapped contour surface chart for the z's corresponding to a set of (x, y).

GeneratedDelaunaySurface

1. Construct a OpenGL 3D Delaunay surface chart for the z's corresponding to a set of (x, y).

Histogram

1. Construct a OpenGL 3D histogram for the z's corresponding to a set of (x, y).

MultiColorScatter

1. Construct a 3D depth color-mapped multi-color scatter surface chart for the z's corresponding to a set of (x, y).

Scatter4D

1. Construct a 3D depth color-mapped multi-color scatter surface chart for the z's corresponding to a set of (x, y).

Package org.drip.curve.calibration

Bootstrapper

1. Interface defining the core bootstrapping methods – setting/bumping specific nodes, setting flat values across all nodes, retrieving specific/collective instrument/node quotes.

ComponentCalibrator

1. Interface defining the curve calibration methods – bootstrapping the discount rate and the hazard rate from the individual component quotes. Calibration can be node-by-node (true bootstrapping) or flat.

ComponentCalibratorBracketing

1. Uses a bracketing technique to find the discount rate/hazard rate. Calibration produces either the implied piece-wise constant forward or the flat root across all the nodes.

ComponentCalibratorNR

1. Uses either the Newton-Raphson method or the Brent's secant method to calibrate the discount rate/hazard rate. Calibration produces either the implied piece-wise constant forward or the flat root across all the nodes.

CreditCurveScenarioGenerator

1. Contains the credit calibration instruments to be used with the component calibrator to produce scenario credit curves.
2. Calibrate and create a bootstrapped or flat credit curve from valuation parameters, recovery rate, discount curves, and fixings.
3. Calibrate an array/tenor map of bootstrapped or flat tenor bumped credit curves from valuation parameters, recovery rate, discount curves, and fixings.

IRCurveScenarioGenerator

1. Holds the IR calibration instruments to be used with the component calibrator to produce scenario discount curves.
2. Calibrate and create a bootstrapped or flat discount curve from valuation parameters, treasury/EDF curves, and fixings.
3. Calibrate an array/tenor map of bootstrapped or flat tenor bumped discount curves from valuation parameters, treasury/EDF curves, and fixings.

Package org.drip.param.config

XMLConfigReader

1. Parses the XML configuration file and extracts the information, tag pairs – such as holiday sets for different locations, logger location, analytics server connection strings, database server connection strings.
2. Loads holiday sets from either directly from the configuration files, or from database setting.

Package org.drip.param.market

BasketMarketParamRef

1. Interface to provide stubs for component IR and credit curves that constitute the basket.

BasketMarketParams

1. Implementation of BasketMarketParamsRef for a specific scenario. Contains maps holding named discount curves, named credit curves, named treasury quote, named component quote, and fixings object.
2. Ability to add and retrieve a named discount curve and a named credit curve.
3. Builds a ComponentMarketParams object from a component given its ComponentMarketParamsRef object.
4. Implements the serialization interface to serialize the BasketMarketParams object instance into byte arrays, as well as de-serialize and populate a BasketMarketParams instance from an input byte array.

ComponentMarketParamRef

1. Interface to provide stubs for component name, IR curve, credit curve, TSY curve, and EDSF curve needed to value the component.

ComponentMarketParams

1. Implementation of the ComponentMarketParamsRef interface. Placeholder for the market parameters needed to value the component object – discount curve, treasury curve, EDSF curve, credit curve, component quote, treasury quote map, and fixings map.
2. Implements the serialization interface to serialize the ComponentMarketParams object instance into byte arrays, as well as de-serialize and populate a ComponentMarketParams instance from an input byte array.

CreditCurveScenarioContainer

1. Placeholder for the bump parameters and the curves for the different credit curve scenarios. Contains the spread and the recovery bumps, and the credit curve scenario generator object that wraps the calibration instruments.
2. Contains the base credit curve, spread bumped up/down credit curves, recovery bumped up/down credit curves, and the tenor mapped up/down credit curves.
3. Depending upon the scenario creation mode, cooks the curves that correspond to the scenarios above, and retrieves them.

IRCurveScenarioContainer

1. Placeholder for the different IR scenario curves. Contains the IR curve scenario generator object that wraps the calibration instruments.
2. Contains the base IR curve, spread bumped up/down IR curves, and tenor mapped up/down credit curves.
3. Depending upon the scenario creation mode, cooks the curves that correspond to the scenarios above, and retrieves them.

MarketParamsContainer

1. Placeholder for the comprehensive suite of the market set of curves for the given date. Contains treasury quote map, fixings map, IR scenario curve map, credit curve scenario set map, and component quote map.
2. Sets/gets scenario discount curve set and scenario credit curve set for a given IR currency/credit curve.
3. Sets/gets quote for a given treasury benchmark or for a full set of treasury benchmarks.
4. Get/set the fixings map.
5. Sets/gets quote for a given component or for a full set of components.
6. Get component market parameters for the given component and scenario.
7. Get the map of tenor component market parameters for the given component across each of the IR tenors.
8. Get the map of tenor component market parameters for the given component across each of the credit tenors.
9. Get the basket market parameters for the given basket and scenario.
10. Get the map of flat IR bumped basket market parameters for the given basket across each of the IR curves.
11. Get the map of flat credit bumped basket market parameters for the given basket across each of the credit curves.
12. Get the map of flat recovery bumped basket market parameters for the given basket across each of the credit curves.
13. Get the double map of the tenor IR bumped basket market parameters for the given basket across each of the IR curves and their tenors.
14. Get the double map of the tenor credit bumped basket market parameters for the given basket across each of the credit curves and their tenors.
15. Get the double map of the tenor recovery bumped basket market parameters for the given basket across each of the credit curves and their tenors.

NodeTweakParams

1. Placeholder for the scenario tweak mode, for either the whole curve, or for segments of it.
2. Contains the tweak type (parallel/proportional), tweak node, or tweak amount.
3. Implements the serialization interface to serialize the NodeTweakParams object instance into byte arrays, as well as de-serialize and populate a NodeTweakParams instance from an input byte array.

Package org.drip.param.pricer

PricerParams

1. Placeholder for the pricing parameters across all product classes and pricing methods. Contains the loss step discretization scheme, time domain unit size, whether survival is to be calculated to the period accrual date or period pay date, and whether current pricing is a calibration operation or not.
2. Implements the serialization interface to serialize the PricerParams object instance into byte arrays, as well as de-serialize and populate a PricerParams instance from an input byte array.

CalibrationParams

1. Placeholder for the calibration parameters – the measure to be calibrated, and the type and the nature of the calibration to be done, and the exercise parameters to which the calibration is set.
2. Implements the serialization interface to serialize the CalibrationParams object instance into byte arrays, as well as de-serialize and populate a CalibrationParams instance from an input byte array.

Package org.drip.param.product

BondCFTerminationParams

1. Contains the termination static/status parameters for the bond. Specifically, indicates whether the bond is perpetual, has been called, or has defaulted.
2. Implements the serialization interface to serialize the BondCFTerminationParams object instance into byte arrays, as well as de-serialize and populate a BondCFTerminationParams instance from an input byte array.

BondCouponParams

1. Contains the coupon parameters for the bond. Indicates whether the bond coupon/spread is, contains the coupon schedule and the coupon type.
2. Implements the serialization interface to serialize the BondCouponParams object instance into byte arrays, as well as de-serialize and populate a BondCouponParams instance from an input byte array.

BondCurrencyParams

1. Contains the currency parameters for the bond. Contains the trade, the coupon, and the redemption currencies.
2. Implements the serialization interface to serialize the BondCurrencyParams object instance into byte arrays, as well as de-serialize and populate a BondCurrencyParams instance from an input byte array.

BondFixedPeriodGenerationParama

1. Contains the period generation parameters for the bond. Contains the frequency, accrual/coupon day-count conventions, effective, maturity, and final maturity dates, and maturity type. The coupon periods generated are and kept in a list.
2. Implements the serialization interface to serialize the BondFixedPeriodGenerationParams object instance into byte arrays, as well as de-serialize and populate a BondFixedPeriodGenerationParams instance from an input byte array.

BondFloaterParams

1. Placeholder for the bond's floating rate parameters. Contains the floater flag, the rate index, the index spread, floater day count, and the current coupon.
2. Implements the serialization interface to serialize the BondFloaterParams object instance into byte arrays, as well as de-serialize and populate a BondFLoaterParams instance from an input byte array.

BondIdentiferParams

1. Placeholder for the bond's identifier parameters. Contains the ISIN, CUSIP, bond ID, and the ticker.
2. Implements the serialization interface to serialize the BondIdentifierParams object instance into byte arrays, as well as de-serialize and populate a BondIdentifierParams instance from an input byte array.

BondIRValuationParams

1. Placeholder for the bond's IR valuation parameters. Contains the bond's IR curve, the quoting convention, the calculation type, the first settle date, and the redemption value.
2. Implements the serialization interface to serialize the BondIRValuationParams object instance into byte arrays, as well as de-serialize and populate a BondIRValuationParams instance from an input byte array.

BondNotionalParams

1. Placeholder for the bond's notional parameters. Contains the bond's current notional and the outstanding notional schedule.
2. Implements the serialization interface to serialize the BondNotionalParams object instance into byte arrays, as well as de-serialize and populate a BondNotionalParams instance from an input byte array.

BondPeriodGenerationParams

1. Placeholder for the generic bond's period generation parameters. Contains the bond's date adjustment parameters for period start/end, period accrual start/end, effective, maturity, pay and reset, first coupon date, and interest accrual start date.
2. Validation results in the generation of the list of periods according to the date generation rules – invalid/inconsistent parameters result in no such list being created.
3. Implements the serialization interface to serialize the BondPeriodGenerationParams object instance into byte arrays, as well as de-serialize and populate a BondPeriodGenerationParams instance from an input byte array.

BondTSYParams

1. Placeholder for the bond's treasury related parameters. Contains the bond's treasury benchmark set, government treasury benchmark, and EDSF short-end benchmark identifiers.
2. Implements the serialization interface to serialize the BondTSYParams object instance into byte arrays, as well as de-serialize and populate a BondTSYParams instance from an input byte array.

CDXIdentifier

1. Placeholder for the identifier parameters for a given standard CDX contract. Contains the index, the tenor, the series, and the version for the given contract.
2. Implements the serialization interface to serialize the CDXIdentifier object instance into byte arrays, as well as de-serialize and populate a CDXIdentifier instance from an input byte array.

CurrencyPair

1. Placeholder for the currency pair object for a given FX contract. Contains the numerator, the denominator, and the quoting currencies, and the PIP factor for the given contract.
2. Implements the serialization interface to serialize the CurrencyPair object instance into byte arrays, as well as de-serialize and populate a CurrencyPair instance from an input byte array.

EmbeddedOptionSchedule

1. Placeholder for the embedded option schedule for the bond. Contains the schedule of exercise dates and factors, the exercise notice period, and the option is to call or put.

Further, if the option is of the type fix-to-float on exercise, contains the post-exercise floater index and floating spread.

2. If the exercise is not discrete (American option), the exercise dates/factors are discretized according to a pre-specified discretization grid.
3. Helper functions help create the schedule from string date/factor arrays.
4. Retrieves whether the option is for a put, whether it is fix-to-float on exercise, and the date/factor array.
5. Implements the serialization interface to serialize the EmbeddedOptionSchedule object instance into byte arrays, as well as de-serialize and populate a EmbeddedOptionSchedule instance from an input byte array.

FactorSchedule

1. Place holder for factor index schedule of all types – in particular coupon/principal factors. Contains equal date/factor arrays.
2. Helper functions create FactorSchedule from arrays of dates/factors, string arrays of dates/factors, and even bullet schedules.
3. Retrieve the factor/factor index for a given date, as well as the array of factors/dates.
4. Gets the effective time-weighted factor between two arbitrary dates.
5. Implements the serialization interface to serialize the FactorSchedule object instance into byte arrays, as well as de-serialize and populate a FactorSchedule instance from an input byte array.

CompCRValParams

1. Placeholder for the credit component's credit valuation parameters. Contains the credit curve, component's recovery and whether that is usable, loss pay lag, and whether accrual is to be applied on default.

2. Implements the serialization interface to serialize the CompCRValParams object instance into byte arrays, as well as de-serialize and populate a CompCRValParams instance from an input byte array.

TsyBmkSet

1. Contains the applicable treasury benchmarks on the given bond. Holds the primary treasury benchmark, and none or many secondary treasury benchmarks.
2. Implements the serialization interface to serialize the TsyBmkSet object instance into byte arrays, as well as de-serialize and populate a TsyBmkSet instance from an input byte array.

Package org.drip.param.valuation

CashSettleParams

1. Placeholder for the cash settlement parameters for a given product or a settlement structure. Contains the settle delay, settlement adjustment calendar, and the cash settle adjustment modes.
2. Implements the serialization interface to serialize the CashSettleParams object instance into byte arrays, as well as de-serialize and populate a CashSettleParams instance from an input byte array.

NextExerciseInfo

1. Placeholder for the next exercise information fields for a given product, starting from the given valuation date. Contains the exercise type, the exercise factor, and the exercise date.
2. Implements the serialization interface to serialize the NextExerciseInfo object instance into byte arrays, as well as de-serialize and populate a NextExerciseInfo instance from an input byte array.

QuotingParams

1. Placeholder for the quoting parameters for a given product or a settlement. Contains the quoting yield day count convention, quoting yield frequency, whether EOM adjustment is to be applied or not, the adjustment calendar, and whether the product is spread quoted or price quoted.

2. Implements the serialization interface to serialize the QuotingParams object instance into byte arrays, as well as de-serialize and populate a QuotingParams instance from an input byte array.

ValuationParams

1. Placeholder for the valuation parameters for a given product. Contains the valuation and the cash pay/settle dates.
2. Implements the serialization interface to serialize the ValuationParams object instance into byte arrays, as well as de-serialize and populate a ValuationParams instance from an input byte array.

WorkoutInfo

1. Placeholder for the workout parameters. Contains the date, the factor, and the yield of the workout.
2. Implements the serialization interface to serialize the Workout object instance into byte arrays, as well as de-serialize and populate a Workout instance from an input byte array.

Package org.drip.product.common

CalibratableComponent

1. Abstract class providing implementation of Component interface. Provides stub for getting the component's primary code, and optionally provides multiple secondary codes.
2. It also defines the calibMeasures method – which calculates and returns the requested measure for the given set of inputs.

Component

1. Abstract class extending ComponentMarketParamRef. Provides methods for getting the component's notional, coupon, effective date, maturity date, coupon amount, and list of coupon periods.
2. Calculates the full set of component measures across a set of specified scenarios, and holds the results in ComponentOutput. Retrieves the specified measure for the component.

Package org.drip.product.creator

BondBuilder

1. Contains the suite of helper functions for creating user defined bonds, optionally with custom cash flows and embedded option schedules (European or American).
2. Creates a full featured bond from custom cash flows (coupon/principal flows or amortization/capitalization schedules), simple fixed rate bonds, simple floater bonds, and bonds from explicitly specified coupon flows and principal flows (principal flows can be specified as either principal pay down/up or outstanding notional).

BondProductBuilder

1. Contains the static parameters of the bond product needed for the full bond valuation.
2. Contains the bond identifier parameters (ISIN, CUSIP), the issuer level parameters (Ticker, SPN or the credit curve string), coupon parameters (coupon rate, coupon frequency, coupon type, day count), maturity parameters (maturity date, maturity type, final maturity, redemption value), date parameters (announce, first settle, first coupon, interest accrual start, and issue dates), embedded option parameters (callable, putable, has been exercised), currency parameters (trade, coupon, and redemption currencies), floater parameters (floater flag, floating coupon convention, current coupon, rate index, spread), and whether the bond is perpetual or has defaulted.
3. BondProductBuilder is created from resultset of a bond ref data entry.
4. Gets either the individual fields or the corresponding bond parameter set.
5. Validation ensures consistency of the BondProductBuilder instance.
6. Implements the serialization interface to serialize the BondProductBuilder object instance into byte arrays, as well as de-serialize and populate a BondProductBuilder instance from an input byte array.

BondRefDataBuilder

1. Contains the entire set of static parameters for the bond product.
2. Contains the bond identifier parameters (ISIN, CUSIP, BBG ID, name short name), the issuer level parameters (Ticker, category, industry, issue type, issuer country, issuer country code, collateral type, description, security type, unique Bloomberg ID, long company name, issuer name, SPN or the credit curve string), issue parameters (issue amount, issue price, outstanding amount, minimum piece, minimum increment, par amount, lead manager, exchange code, country of incorporation, country of guarantor, country of domicile, industry sector, industry group, industry sub-group, senior/sub), coupon parameters (coupon rate, coupon frequency, coupon type, day count), maturity parameters (maturity date, maturity type, final maturity, redemption value), date parameters (announce, first settle, first coupon, interest accrual start, next coupon, previous coupon, penultimate coupon, and issue dates), embedded option parameters (callable, putable, has been exercised), currency parameters (trade, coupon, and redemption currencies), floater parameters (floater flag, floating coupon convention, current coupon, rate index, spread), trade status, ratings (S & P, Moody, and Fitch), and whether the bond is private placement, is registered, is a bearer bond, is reverse convertible, is a structured note, can be unit traded, is perpetual or has defaulted.
3. BondRefDataBuilder is created from result-set of a bond ref data entry.
4. Gets either the individual fields or the corresponding bond parameter set.
5. Implements the serialization interface to serialize the BondRefDataBuilder object instance into byte arrays, as well as de-serialize and populate a BondRefDataBuilder instance from an input byte array.

CDXRefDataBuilder

1. Contains the entire set of static and valuation parameters for the named Standard CDX Product.
2. Contains the Index Curve ID, Index Curve SPN, Index Label, Index Name, Index Issue Date, Index Maturity Date, Index Coupon, Index Currency, Index Full First Stub, Index Day Count Convention, Index Recovery, Index Coupon Frequency, Index Reference Entity ID, Index Class, Index Series, Index Group Name, Index Short Group Name, Index Version, Index Life Span, Index Curvy Curve ID, Index Factor, Original Component Count, Defaulted Component Count, Index Location, whether the index pays accrued on default, whether the index knocks out on default, whether the index is quoted as a CDS, Index Bloomberg Ticker, and Index Short name.
3. Individual functions validate each one of the above fields.
4. Finally, it also has functionality to create the actual index basket product (along with the corresponding cash-flows).

Package org.drip.product.credit

BasketBond

1. Placeholder for the bond basket/bond ETF product contract details. Contains the basket name, the basket notional, the component bonds, and their weights.
2. Retrieves the notional at different times, coupon, effective/maturity dates, the flow periods, the component IR and the component credit curves.
3. The value method returns a full set mapped set of all the product measures for a given set of scenario curves. Using this in conjunction with the calcMeasures of the BondBasket produces a comprehensive set of all the mapped scenario measures.
4. Implements the serialization interface to serialize the BondBasket object instance into byte arrays, as well as de-serialize and populate a BondBasket instance from an input byte array.

BasketDefaultSwap

1. Placeholder for the basket default swap product contract details. Contains effective date, maturity date, coupon, coupon day count, coupon frequency, basket components, basket notional, loss pay lag, and optionally the outstanding notional schedule and the flat basket recovery.
2. Helper functions create named CDX from an effective date, maturity date, coupon, IR curve, and a set of components – in a few different forms.
3. Retrieves the notional at different times, coupon, effective/maturity dates, the flow periods, the component IR and the component credit curves.
4. The value method returns a full set mapped set of all the product measures for a given set of scenario curves. Using this in conjunction with the calcMeasures of the BasketProduct produces a comprehensive set of all the mapped scenario measures.

5. Implements the serialization interface to serialize the BasketDefaultSwap object instance into byte arrays, as well as de-serialize and populate a BasketDefautSwap instance from an input byte array.

BasketProduct

1. Abstract class extending BasketMarketParamRef. Provides methods for getting the basket's components, notional, coupon, effective date, maturity date, coupon amount, and list of coupon periods.
2. Calculates the full set of basket measures across a set of specified scenarios, and holds the results in BasketOutput. Retrieves the specified measure for the basket.

Bond

1. Base class that extends CreditComponent abstract class and implements the functionality behind bonds of all kinds.
2. Bond static data is captured in a set of 11 container classes – BondTSYParams, BondCouponParams, BondNotionalParams, BondFloaterParams, BondCurrencyParams, BondIdentifierParams, BondIRValuationParams, CompCRValParams, BondCFTerminationEvent, BondFixedPeriodGenerationParams, and one EmbeddedOptionSchedule object instance each for the call and the put objects. Each of these parameter set can be set separately.
3. Gets the bond's different identifiers – ISIN, CUSIP, primary/secondary codes, and bond's canonical name.
4. Gets the bond's coupon at different times, and the IR/credit/EDSF/TSY benchmark names.
5. Retrieves the notional at different times, effective/maturity dates, coupon/loss flow periods, and component credit valuation parameters.

6. Retrieves floater parameters (rate index, current coupon, float spread, foating coupon convention), embedded option schedule for call/put, whether the bond is amortizing, defaulted, is perpetual, coupon/accrual day-count, coupon type, coupon frequency, trade/redemption/coupon currency.
7. Calculations are done for: reset date for the period given by the valuation date, accrual, theoretical price from a discount/credit curve, price/yield/Z spread/credit basis/treasury spread/G Spread/I Spread/par asset swap spread/duration/convexity from price/yield/Z spread/credit basis/treasury spread/G Spread/I Spread/par asset swap spread inputs. Calculations are done to either a specified work-out date, or maturity/optimal exercise date.
8. The value method returns a full set mapped set of all the bond product measures for a given set of scenario curves. Using this in conjunction with the calcMeasures of the Component produces a comprehensive set of all the mapped scenario measures.
9. Bond also implements calibMeasures method, which calculates and returns the specific measure requested.
10. Implements a calibrator that calibrates the yield, the credit basis, or the Z spread for the bond given the price input. Calibration happens via either Newton-Raphson method, or via bracketing/root searching.
11. Implements the serialization interface to serialize the Bond object instance into byte arrays, as well as de-serialize and populate a Bond instance from an input byte array.

CreditComponent

1. Base abstract class that extends CalibratableComponent over which all credit components are implemented.
2. Creates methods to get the credit valuation parameters, the coupon and the loss flows, and recovery at either a single date, or time weighted average between two dates (component recovery, if available, precedes curve recovery).

CreditDefaultSwap

1. Placeholder for the credit default swap product contract details. Contains effective date, maturity date, coupon, coupon day count, coupon frequency, contingent credit, currency, basket notional, credit valuation parameters, and optionally the outstanding notional schedule.
2. Helper functions create named CDS from an effective date, maturity date/tenor, coupon, IR curve, credit valuation parameters, and comprehensive coupon period generation parameters – in a few different forms.
3. Retrieves the notional at different times, coupon, effective/maturity dates, the flow periods, the component IR/Credit curves, the component credit valuation parameters, the CDS recovery, and the CDS primary/secondary codes.
4. The value method returns a full set mapped set of all the product measures for a given set of scenario curves. Using this in conjunction with the calcMeasures of the Component produces a comprehensive set of all the mapped scenario measures.
5. Implements a flat spread calibrator that calibrates a flat curve off of the current CDS contract from a given market price.
6. Implements the serialization interface to serialize the CreditDefaultSwap object instance into byte arrays, as well as de-serialize and populate a CredirDefaultSwap instance from an input byte array.

StandardCDXManager

1. Placeholder for all the standard CDX/iTRAXX named product details. Contains the named CDX/iTRAXX series identifier (series, version, and index), and maps that retrieve the first coupon date or the series from the other, given an index.
2. Functionality provided retrieves the CDX/iTRAXX index given the index, the series, and the tenor, as well as retrieve the on-the-run index corresponding to the given date and tenor.

3. Finally, it also provides functionality to retrieve the pre-set and pre-loaded CDX names, their first coupon dates, and descriptions.

StandardCDXParams

1. Placeholder for the given standard CDX/iTRAXX index. Contains the components, the currency, and the coupon for the given index.

Package org.drip.product.fx

FXForward

1. Placeholder for the FX forward product contract details. Contains the effective date, the maturity date, the currency pair and the product code.
2. Helper functions create named FXForward object from an effective date, maturity date/tenor, and currency pair.
3. Retrieves the product currency pair, the effective/maturity dates, and the primary/secondary codes.
4. Implies the FX forward as either an outright or a PIP from the input valuation parameters, FX spot, numerator/denominator discount curves based on either the numerator discount curve or the denominator.
5. Calibrates the discount curve basis from the input valuation parameters, FX spot/forward, numerator/denominator discount curves based on either the numerator discount curve or the denominator.
6. Implements a flat spread calibrator that calibrates a flat curve off of the current CDS contract from a given market price.
7. The value method returns a full set mapped set of all the product measures for a given set of scenario curves.
8. Implements a FX basis calibrator that calibrates a basis to either the numerator or the denominator curve off of the current FX forward through the Newton Raphson method.
9. Implements the serialization interface to serialize the FXForward object instance into byte arrays, as well as de-serialize and populate a FXForward instance from an input byte array.

FXSpot

1. Place holder for the FX spot contract details – the spot date and the currency pair.
2. Implements the serialization interface to serialize the FXSpot object instance into byte arrays, as well as de-serialize and populate a FXSpot instance from an input byte array.

Package org.drip.product.quote

ComponentQuote

1. Placeholder for the different types of quotes for a given component. Contains a single market field/quote pair, but multiple alternate named quotes (to accommodate quotes on different measures for the component).
2. Implements the serialization interface to serialize the ComponentQuote object instance into byte arrays, as well as de-serialize and populate a ComponentQuote instance from an input byte array.

LiveQuote

1. Interface exposing the “refresh” method to be implemented by any derived product quote.

Quote

1. Contains the details corresponding to a product quote. Contains the quote value, quote instant for the different quote sides (bid/ask/mid).
2. Implements the serialization interface to serialize the Quote object instance into byte arrays, as well as de-serialize and populate a Quote instance from an input byte array.

Package org.drip.product.rates

Cash

1. Placeholder for the cash product contract details. Contains effective date, maturity date, coupon, coupon day count, coupon frequency, currency, basket notional, and optionally the outstanding notional schedule.
2. Helper functions create Cash instance from effective date, maturity date/tenor, and IR curve.
3. Retrieves the notional at different times, coupon, effective/maturity dates, the flow periods, the IR curves, and the product primary/secondary codes.
4. The value method returns a full set mapped set of all the product measures for a given set of scenario curves. Using this in conjunction with the calcMeasures of the Component produces a comprehensive set of all the mapped scenario measures.
5. Implements the serialization interface to serialize the Cash object instance into byte arrays, as well as de-serialize and populate a Cash instance from an input byte array.

EDFuture

1. Placeholder for the euro-dollar product contract details. Contains effective date, maturity date, coupon, coupon day count, coupon frequency, currency, basket notional, and optionally the outstanding notional schedule.
2. Helper functions create EDF instance from effective date, maturity date/tenor, EDF code, and IR curve, as well create a euro-dollar pack from a given date.
3. Retrieves the notional at different times, coupon, effective/maturity dates, the flow periods, the IR curves, and the product primary/secondary codes.
4. The value method returns a full set mapped set of all the product measures for a given set of scenario curves. Using this in conjunction with the calcMeasures of the Component produces a comprehensive set of all the mapped scenario measures.

5. Implements the serialization interface to serialize the EDFuture object instance into byte arrays, as well as de-serialize and populate a EDFuture instance from an input byte array.

InterestRateSwap

1. Placeholder for the interest-rate-swap product contract details. Contains effective date, maturity date, coupon, coupon/accrual day count, coupon frequency, currency, basket notional, floating rate index, and optionally the outstanding notional schedule.
2. Helper functions create IRS instance from effective date, maturity date/tenor, IRS code, and IR curve, as well create a euro-dollar pack from a given date.
3. Retrieves the notional at different times, coupon, effective/maturity dates, the flow periods, the IR curves, and the product primary/secondary codes.
4. The value method returns a full set mapped set of all the product measures for a given set of scenario curves. Using this in conjunction with the calcMeasures of the Component produces a comprehensive set of all the mapped scenario measures.
5. Implements the serialization interface to serialize the InterestRateSwap object instance into byte arrays, as well as de-serialize and populate a InterestRateSwap instance from an input byte array.

Package org.drip.service.api

FI

1. Class exposes all the DRIP API to clients – this class is the main functional interface. Retains an instance of the current date/time, bond valuation data and bond ref data caches, and SQL connections to all the databases.
2. Init initializes the holiday calendars, data-base connections, and the current date/times.
3. Gets all the holiday locations, available day counts, week day/weekend/combined holidays for the location set and year, the weekend days, and whether the given day is a holiday or not.
4. Calculates the year fraction contained in the start/end dates, day count convention, and holiday calendar set; also adjusts/rolls the given date to the next business day in accordance with the holiday calendar set and adjustment convention.
5. Gets the set of on-the-run treasury ISINs and their yields.
6. Creates a component from the code, and makes a component set.
7. Retrieves the available IR/TSY curve names, loads the live/EOD IR/TSY curve for a given EOD or a set of EODs.
8. Loads the live/EOD Cash/EDF/IRS curve(s) for a given EOD or a set of EODs.
9. Retrieves the available CDS curve names, loads the live/EOD CDS/bond/combined credit curve for an issuer for a given EOD or a set of EODs.
10. Add/remove/retrieve bond/bond ref data from bond ID (ISIN/CUSIP etc)
11. Get the bond's call/put schedules (optionally, given a date)
12. Create a fixing for the given bond at the given period, get all available tickers, and get all the ISINs for the given ticker, and their outstanding notional.
13. Get outstanding issuer notional aggregated by pre-specified maturity buckets.
14. Calculate the bond's price/yield, theoretical credit price, spread to treasury benchmark, Z Spread, I Spread, par asset swap spread, and credit basis given the bond's yield/price.

15. Get the bond’s named boolean, string, integer, double, or date field.
16. Create a CDX product given the effective and the maturity dates.

Package org.drip.service.env

BondManager

1. Container that holds the EOD and bond static information for the given issuer.
2. Holds bond EOD bid/ask marks, ticker-wise maturity sorted set of bonds, map containing bond instance to ID (ISIN/CUSIP), flags for different formats of runs and corresponding displays/calculations.
3. Loads the option schedule for all bonds.
4. Calculates the bond measures and generates the formatted runs for a pair of bid/ask price for the given bond.
5. Generates formatted runs of measures for all the bonds for a given EOD.
6. Retrieves and/or loads the bid/ask/mid levels for a single bond or for all the bonds on a given EOD.
7. Builds and retrieves the bond object associated with a ResultSet and a MarketParamsContainer object.
8. Loads the bond from the given bond ID, MarketParamsContainer, the SQL statement, and an optional date to determine when the discretization of an American option should start.
9. Loads the entire ref data for a bond given the ID and the SQL statement.
10. Gets all the available issuer tickers in the database.
11. Retrieves all the available ISINs for the given issuer ticker.
12. Loads al the bonds and commits them to memory.
13. Calculates the measures for all the bonds associated with a ticker maturity after the valuation parameters, given the bid/ask prices and the market parameters.
14. Calculates the measures for all the bonds associated with a ticker maturity after the valuation parameters, given the closing bid/ask prices and the closing market parameters.
15. Calculates and saves/loads all the bond measures for all the bonds corresponding to the given closing date.

CDSManager

1. Container that holds the EOD and CDS/credit curve information on a per-issuer basis.
2. Retrieves all the credit curves available for the given EOD from the database.
3. Saves the EOD credit measures that correspond to the credit curve represented by the SPN ID from the input market parameters and the EOD tenor quotes from the database.
4. Save al the EOD credit measures all the curves in the given MPC.
5. Load all the credit curves corresponding to a given EOD from the database.

EnvManager

1. Class sets the environment/connection parameters, and populates the market parameters for the given EOD.

EODCurves

1. Container that exposes the functionality to create the set of closing IR and credit curves for a given EOD.
2. Builds the named EOD credit curve for the given SPN ID/discount curve combination by retrieving the EOD quotes from the database, and using the specified discount curve.
3. Builds the named EOD IR curve of a specific type (cash/EDF/IRS) for the given discount curve name and currency by retrieving the EOD quotes from the database.
4. Builds the named full EOD IR curve using instruments of all types (cash/EDF/IRS) for the given instrument set type (swaps/treasuries), discount curve name, and currency by retrieving the corresponding EOD quotes from the database.

5. Loads the requested type of discount curve (cash/EDF/IRS/full/TSY) for the given currency and the EOD, and optionally adds it to the market params container.

RatesManager

1. Gets the discount curve names of the requested type (or the full discount curve) for the given EOD.
2. Loads the full closing discount curve for the given EOD.

StaticBACurves

1. Class to create a set of discount/credit curves from made up marks for a given EOD.
2. Creates a set of USD treasury bonds/ED futures/IR swaps and their quotes, builds a discount curve from them, and loads it to the market parameters container.
3. Creates a set of credit default swaps and their quotes, builds a credit curve from them, and loads it to the market parameters container.

Package org.drip.service.external

AnalyticsClient

1. Class that captures the requests for the FI server from the GUI client, formats them, and sends them to the AnalyticsServer.

AnalyticsServer

1. Class that receives the requests from the analytics client, and invokes the FI functionality, and sends the client the results.

Package org.drip.service.sample

BondAnalyticsAPISample

This class contains a demo of the bond analytics API sample. It has functions illustrating the creation and the usage of the principal and coupon schedules, custom fixed, floaters, and custom bonds from arbitrary cash flows, bond with option schedules, run through of the full set of bond relative value analytics, and calibration of credit curves from CDS/bond prices and their corresponding measures.

BondBasketAPISample

This class contains a demo of the bond basket API sample. It has functions illustrating the creation and the usage of the bond basket from its components, and a run through of the full set of bond basket value.

BondLiveAndEODAPISample

This class contains a demo of the live and EOD API calls for bonds. It displays the full set of RV, interest rate, and credit measures for a given bond identifier (ISIN/CUSIP) given closing price/yield/spread to treasury, full set of RV measures for all the bonds in a given ticker, their individual and bucketed outstanding notional, full set of valuation and relative value measures, coupon flows, and loss flows for a given identifier.

BondStaticAPISample

This class contains a demo of the API calls for bond static fields. It displays the full set of static fields for a given bond identifier (ISIN/CUSIP).

CDSAnalyticsAPISample

This class contains a demo of the CDS analytics API sample. It has functions illustrating the creation and the usage of CDS, display of their coupon and loss flows, and calibration of credit curves from CDS prices of all types, and their corresponding measures, as well as from the survival probabilities and hazard rates.

CDSBasketAPISample

This class contains a demo of the CDS basket API sample. It has functions illustrating the creation and the usage of pre-set and pre-loaded CDS basket, the full set of named CDX indices and their descriptions, and a run through of the full set of bond basket value.

CDSLiveAndEODAPISample

This class contains a demo of the live and EOD API calls for CDS. It displays the full set of RV, interest rate, and credit measures for a given CDS given several forms of CDS quotes, and the creation of the calibrated credit curve for the given CDS curve name and EOD.

DayCountAndCalendarAPISample

This class contains a demo of the day count and the calendar analytics API sample. It has functions display of the built-in holiday locations, week days, weekends, and holiday sets

between a start date and an end date for a set of holiday calendars, available day count calendars, and samples to adjust/roll according to holiday sets.

FXAPISample

This class contains a demo of the FX analytics API sample. It has functions to create and use FX spot, FX forward curves (as either outright or as PIP), create basis curves both as spot-starting basis as well as forward basis to the domestic/foreign rates curves, and recovers the forward curve (in differing formats) from the corresponding basis.

RatesAnalyticsAPISample

This class contains a demo of the rates analytics API sample. It has functions illustrating the creation and the usage of different rates instruments (cash/futures/swaps), display of their coupon flows, and calibration of rates curves from discount factors and forward rates, and their corresponding measures.

RatesLiveAndEODAPISample

This class contains a demo of the live and EOD API calls for rates instruments. It displays the full set cash/EDF/swaps/treasury/composite types of rates curves available on a EOD, their calibration, creation and usage.

Package org.drip.service.external

FIGen

1. Collection of the static generic utility functions used by the internal and external DRIP modules.
2. Initializes DRIP<->Bloomberg holiday code map, and rates curve switcher map.
3. Utility functions to parse Bloomberg holiday codes, and the get tenor/month code from frequency.
4. Construct the default rate index string from the currency and the coupon frequency.
5. Create a DRIP Julian Date from java date.
6. Convert string to Boolean.
7. Make Oracle date from YYYYMMDD string/ Bloomberg date string.
8. Make double array from string tokenizer.
9. Format input number into different requested formats (with optional padding).
10. Create a fixings object for the bond, value date, and the fix coupon.
11. Generate a GUID string.
12. Generate the credit products loss period in accordance with the parameters specified in the pricer.

Validatable

1. Interface that exposes the “validate” method – needed to validate the state of the implementing object.

Package org.drip.util.date

DateTime

1. Complete representation of the instantiation-time date and time objects. Retrieves the date and time.
2. Implements the serialization interface to serialize the DateTime object instance into byte arrays, as well as de-serialize and populate a DateTime instance from an input byte array.

JulianDate

1. Comprehensive representation of the date and date manipulation functionality. Canonical representation is as a Julian double.
2. Converts year, month, and day to Julian double.
3. Converts the Julian double to Y/M/D string.
4. Extracts the year, month, or day from the input Julian double.
5. Computes days elapsed in the year, days remaining in the year, or is the current day a leap year.
6. Finds out how many “leap days” are between two given dates – right/left inclusive.
7. Characters corresponding to the month/day in a few formats (regular/Oracle), and vice versa.
8. Number of days in the given month, whether the given Julian double corresponds to the end-of-month.
9. Different ways of constructing the Julian date object – Today, from YMD, from DDMMMYYYY, or from simply a Julian double.
10. Gets the object’s Julian double.
11. Add/subtract days/business days/months/years/tenor.
12. Gets the first EDF start date (based on the EDF roll months).

13. Difference to/equal to/compare with another date,
14. Get Oracle string representation, and the hash code.

Package org.drip.util.internal

FIUtil

1. Utility functions meant primarily for internal DRIP modules.
2. Calculates the yield DF given the yield and time fraction, and the treasury benchmark string from the valuation date and the maturity date.

Logger

1. Implements level-set logging, backed by either the screen or a file. Logging always includes time-stamps, and happens according to the level requested.