Credit Trader Suite User/Developer Guide

Lakshmi Krishnamurthy

v1.6, 19 July 2012

Introduction

Credit Trader Suite of libraries aims to provide open source analytics and trading/valuation system solution suite for credit and fixed income products. To this end, it implements its functionality over 3 main libraries – CreditProduct, CreditAnalytics, and RegressionSuite.

Overview and Problem Space Coverage

The main challenges that Credit Trader Suite attempts to address are:

- Implementation of day count conventions, holidays calendars, and rule-based period generators
- Abstract the functionality behind curves, parameters, and products onto defined interfaces
- Unified/standardized methods for curve calibrations, parameter and product implementers and constructors
- Environmental system to hold live ticks, closing marks, and reference data containers
- Enhanced analytics testing
- Ease of usage, installation, and deployment

While a number of other libraries - both Open Source implementations such as Quantlib and its variants and proprietary systems such as Fincad, NumeriX, Algorithmics, they typically cater to the needs of the wider financial mathematics community, thus diluting their value towards treating credit products. Further, few of them inherently export a curve/product/product models that work well with products quotes, marks, and reference data sources, thereby forcing development teams to build their own integration layers from scratch. Finally, building the components of functional credit-trading system requires additional layers of development over analytics.

Credit Trader Suite is an attempt to overcome these shortcomings. It aims to bring the aspects mentioned above together in one Open Source implementation that readily integrates onto existing systems.

Main Libraries and their Purpose

The libraries the constitute the Credit Analytics Suite are:

- CreditProduct – Focused on the core analytics, and the curve, the parameter, and the product definitions
- CreditAnalytics – Concerned with the construction and the implementation of the interfaces defined in CreditProduct, analytics environment management, and functional distribution
- RegressionSuite – This aims to ease testing of analytics, measurement and generation of the execution time distribution, as well as release performance characterization.

CreditProduct Description and Problem Space Coverage

CreditProduct aims to define the functional and behavioral interfaces behind curves, products, and different parameter types (market, valuation, pricing, and product parameters). To facilitate this, it implements various day count conventions, holiday sets, period generators, and calculation outputs.

CreditProduct library achieves its design goal by implementing its functionality over several packages:

- Dates and holidays coverage: Covers a variety of day count conventions, 120+ holiday locations, as well as custom user-defined holidays
- Curve and analytics definitions: Defines the base functional interfaces for the variants of discount curves, credit curves, and FX curves

- Market Parameter definitions: Defines quotes, component/basket market parameters, and custom scenario parameters
- Valuation and Pricing Parameters: Defines valuation, settlement/work-out, and pricing parameters of different variants
- Product and product parameter definitions: Defines the product creation and behavior interfaces for Cash/EDF/IRS (all rates), bonds/CDS (credit), and basket bond/CDS, and their feature parameters.
- Output measures container: Defines generalized component and basket outputs, as well customized outputs for specific products

CreditAnalytics Description and Problem Space Coverage

CreditAnalytics provides the functionality behind creation, calibration, and implementation of the curve, the parameter, and the product interfaces defined in CreditProduct. It also implements a curve/parameter/product/analytics management environment, and has packaged samples and testers.

CreditAnalytics library achieves its design goal by implementing its functionality over several packages:

- Curve calibration and creation: Functional implementation and creation factories for rates curves, credit curves, and FX curves of al types
- Market Parameter implementation and creation: Implementation and creation of quotes, component/basket market parameters, as well as scenario parameters.
- Product implementation and creation: Implementation and creation factories for rates products (cash/EDF/IRS), credit products (bonds/CDS), as well as basket products.
- Reference data/marks loaders: Loaders for bond/CDX, as well a sub-universe of closing marks
- Calculation Environment Manager: Implementation of the market parameter container, manager for live/closing curves, stub/client functionality for serverization/distribution, input/output serialization.

- Samples: Samples for curve, parameters, product, and analytics creation and usage
- Unit functional testers: Detailed unit scenario test of various analytics, curve, parameter, and product functionality.

RegressionSuite Description and Problem Space Coverage

RegressionSuite aims to incorporate measurement of the startup lag, measurement of accurate execution times, generating execution statistics, customized input distributions, and processable regression specific details as part of the regular unit tests.

RegressionSuite library achieves its design goal by implementing its functionality over several packages:

- Regression Engine: Provides control for distribution set, invocation strategy, and load.
- Unit Regression Executor: Framework that implements set up and tear-down, as well as generate run details
- Regression Statistics: Execution time distribution, start-up and other event delay measurements, and system load monitoring
- Regression Output: Fine grained regressor level output, module aggregated output, sub-element execution time estimation.
- Regressor Set: Module containing set of regressors, group level turn on/off and execution control
- Regression Utilities: Formatting and tolerance checking.

Design Objectives

This section covers the design objectives across several facets – functional, software, system, usage, and deployment aspects.

Financial Feature design attributes

The chief design aims from a financial functionality angle are:

- Interface representations of curve, parameter, and products
- Separation of the creation and the implementation modules from the exposed functional interface behavior
- Re-usable functional and behavioral abstractions around financial products
- Provide “open” public implementations of the standard analytics functionality such as day count conventions, holidays, date representations, rule based period generation etc
- Abstraction of the quote, the market parameter and the pricing structures
- Abstraction and implementation of the standard curve calibration

Software Feature design attributes

The chief design aims from a software design angle are:

- Logical functionality identification/localization and functional group partitioning
- Clearly defined interface structure
- Implementation and creation factory bodies
- Reach and interaction through interfaces only

System Feature Design Attributes

The key system design aims are:

- Functionality needs to be readily serverizable and distributable
- Provide built in serialization, marshalling, and persistence of all the main components
- Management containers around the products, the curves, and the parameter sets, and establishing the execution control environment

Analytics Usage Design Objectives

The key usage design goals are:

- The analytics modules should provide comprehensive credit product risk, valuation, and pricing functionality from a set of functional API
- Ease of use
- Flexible
- When direct object access is needed, use only through the object model interface (and amend the interface as appropriate)

Test Design Objectives

The key testing design goals in this area are:

- Comprehensive unit testing of curve, parameters, and product implementation
- Extensive composite scenario testing
- Environment and server components testing
- Release time performance characterization and execution time and execution resource statistics calculation

Installation, Dependency, and Deployment Design Objectives

The key design goals in this area are:

- Minimize dependency on external modules
- Ease of installation and deployment
- Customizability – for non-standard setups – through the supplied configuration file.

Credit Product

Credit Product Library consists of the following 14 packages:

1. Date & Time Manipulators: This contains functionality for creating, manipulating, and adjusting dates, as well as time instants (to nano-second granularity).

2. Day-count Parameters, Conventions, and Date Adjustment Operations: This contains the functionality for day count generation and date adjustment according to specific rules. It also holds parameters needed for specific day count conventions.

3. Location Specific Standard Holiday Set: This contains all the non-weekend holidays that correspond to a specific location jurisdiction, and its description. Each location implements it holidays in a separate class.

4. Custom Holidays: This provides the ability to specify custom holidays, if the standard ones provided earlier are insufficient. Different types of holidays can be added – variable, fixed, static, as well as weekends for a given location.

5. Curve Analytics Definitions: This provides the definition of all the curve objects – the base curve, the rates curves (discount curves and zero curves), credit curves, and the FX curves (FX basis and FX forward curves).

6. Cash flow Period: This contains the cash flow period functionality, as well as place holders for the period related different curve factors.

7. Analytics Support Utilities: This contains utility functions for manipulating the core Credit Product modules, generic utility functions, and a logger.

8. Quotes, Market, and Scenario Parameters Definitions: This contains the classes that implement the definitions for all parameters except product feature parameters – quotes, calibration parameters, market parameters, tweak parameters, and the scenario curves.

9. Pricer Parameters: This contains the pricing parameters corresponding to a given product and model.

10. Valuation Parameters: This contains all the non-market and non-product parameters needed for valuing a product at a given date.

11. Product Definitions: This contains interface definitions for all products, along with definitions for credit, rates, and FX components and specific credit/rates/FX products, and baskets.

12. Product Parameters: This contains the implementations of the features required for a complete construction of an instance of the product.

13. Product RV and Batch Calculation Outputs: This contains the bulk results of pricing and relative value calculation for the products.

14. Serializer: This interface defines the core object serialization methods – serialization into and de-serialization out of byte arrays, as well as the object serializer version.

Credit Product: Date Time Manipulators

Date Time Manipulators are implemented in the package *org.drip.analytics.date*. It contains functionality for creating, manipulating, and adjusting dates, as well as time instants (to nano-second granularity).

The functionality is implemented in 2 classes: DateTime and JulianDate, and both are serializable.

JulianDate

JulianDate contains the representation and functionality behind the date implementation. Functionality it provides helps:

- Date creation (today, from YMD, different string representations)
- Date comparison (difference, number of leap days)
- Generate date by adjusting an existing date by a specified period rule and a holiday calendar
- Date Characterization – (the year/month/day, leap year, EOM, days elapsed/remaining in the current year)
- Product specific date generation – next EDF start date, IMM start date.

DateTime

DateTime contains nano-second level time snap of date/time instant.

Credit Product: Day Count Parameters, Conventions, and Date Adjustment Operations

Day Count Calculators are implemented in the package *org.drip.analytics.daycount*. It contains the functionality for day count generation and date adjustment according to specific rules. It also holds parameters needed for specific day count conventions.

The functionality is implemented across 3 classes: ActActDCParams, Convention, and DateAdjustParams.

ActActDCParams

This class contains the start and the end dates of the reference period corresponding to the Act/Act convention, as well as the reference frequency.

Convention

This class contains a collection of static methods that provide holiday and day-count related functionality. Following is the set of functionality implemented:

- Date rolls/adjust: Actual, Following, Modified following, previous, and modified previous.
- Locale specific holidays: Access to the full set of non-weekend holidays for a given locale. Also lists the weekend days corresponding to a locale.
- Day Count Year Fraction: Implementation of over 80+ different day count conventions – please consult Credit Analytics site for the implemented day counts.

DateAdjustParams

This class contains the holiday calendar and the roll mode corresponding to a date roll specification. For possible roll modes, please consult Convention.

Credit Product: Location Specific Standard Holiday Set

Location Specific Holidays are implemented in the package *org.drip.analytics.holset*. It contains all the holidays that correspond to a specific location jurisdiction, and its description.

The functionality is implemented in its own location qualified class instance - each of which is an instance of the LocationHoliday interface.

LocationHoliday

This base interface - implemented by all the locale-specific holidays – provides methods to identify the location by name, as well as to retrieve the full set of non-weekend holidays.

Other classes in this package provide explicit holidays and the locale name. So far, Credit Product has about 130 locales implemented – please consult the Credit Analytics site for what they are.

Credit Product: Custom Holidays

Custom Holiday creators are implemented in the package *org.drip.analytics.holiday*. It provides the ability to add holidays, it the standard ones provided earlier are insufficient. Different types of holidays can be added – variable, fixed, static, as well as weekends for a given location.

Different holiday types are implemented in their own classes – they are Static, Fixed, and Variable, each of which extends the Base holiday class. Weekend is implemented in a separate class. All holiday instances for a given locale are maintained on a named holiday container.

Base

This base abstract custom holiday class is implemented by all the other custom holiday implementations. It contains the holiday descriptions and provides functionality to roll the holiday. It also exposes the stub for realizing a holiday rule into an actual date for the given year.

Fixed

This custom holiday class implements the holiday rule: *25th of December, adjusted for weekends*.

Locale

This custom holiday class contains the weekend and the custom holiday set.

Static

This class implements a fully specified date, e.g., *1 January 2012*. No adjustments are applied.

Variable

This custom holiday class implements the rule: *2nd last Thursday of November, adjusted for weekends*.

Weekend

This custom holiday class contains the array of weekend days.

Credit Product: Curve Analytics Definitions

Curve Analytics Curve definitions are implemented in the package *org.drip.analytics.definition*. It provides the definition of all the curve objects – the base curve, the rates curves (discount curves and zero curves), credit curves, and the FX curves (FX basis and FX forward curves).

Functionality in this package is implemented across 6 classes – Curve, DiscountCurve, ZeroCurve, CreditCurve, FXBasisCurve, and FXForwardCurve.

CreditCurve

This interface exposes the following Credit Curve functionality:

- Instruments: Sets/retrieves the calibration instruments
- Curve re-generation: Generates a parallel hazard shifted or flat hazard credit curve
- Credit Curve implied: Retrieves survival factor, hazard rate, or recovery rate to a specified date, or averaged over a set of dates.

Curve

This is the base interface across implemented by all curves, and exposes the following functionality:

- Setting Node Value: Sets/bumps the value in a specified curve node, or sets a flat value across all nodes.
- Calibration Components: Retrieves the full set of the calibration components and their quotes.

- Curve re-generation: Creates a curve object from a specified parallel shift, or scenario tweak parameters.

DiscountCurve

This interface exposes the following Discount Curve functionality:

- Instruments: Sets/retrieves the calibration instruments
- Curve re-generation: Generates a parallel hazard shifted or flat hazard discount curve
- Credit Curve implied: Retrieves the discount factor of the implied rate to a specified date, or averaged over a set of dates.

FXBasisCurve

This interface exposes the FX Basis Curve functionality. It retrieves the CurrencyPair, the FX Spot, and the array of the full set of FX forwards.

FXForwardCurve

This interface exposes the FX Forward Curve functionality. It retrieves the CurrencyPair, the FX Spot, and provides the functionality for array of the full set FX basis to a pair of discount curves, as well as the bootstrapped basis.

ZeroCurve

This interface calculates the zero rate to a given date.

Credit Product: Cash flow Period

Cash flow period functionality is implemented in the package *org.drip.analytics.period*. It contains the cash flow period functionality, as well as place holders for the period related different curve factors.

Functionality in this package is implemented across 4 classes – Period, CouponPeriod, CouponPeriodCurveFactors, and LossPeriodCurveFactors.

CouponPeriod

This class extends the Period class, and contains the reset and the year fraction generation parameters. It also provides static methods to generate a set of periods going forward/backward according to the specified set of date rules.

Period

This class contains the period start/end date, the period accrual start/end date, the pay date, and the period year fraction.

CouponPeriodCurveFactors

This class enhances the Period class with the starting/ending discount factors, survival factors, and notionals.

LossPeriodCurveFactors

This class enhances the Period class with the starting/ending survival factors, as well as the effective discount factor, recovery, and notional over the period.

Credit Product: Analytics Support Utilities

Analytics Support functionality is implemented in the package *org.drip.analytics.support*. It contains utility functions for manipulating the core Credit Product modules, generic utility functions, and a logger.

Functionality in this package is implemented across 3 classes – AnalyticsHelper, GenericUtil, and Logger.

AnalyticsHelper

This class contains numerous static functions that link the different modules of Credit Product together – helper functions that tie the curve, product, parameters, periods, dates, and other classes with each other.

GenericUtil

This class contains assorted utility functions used by Credit Analytics. These functions do not reference other Credit Analytics objects.

Logger

This class logs time-stamped log messages, and is used throughout Credit Analytics.

Credit Product: Quote, Market, and Scenario Parameters

Quote, Market, Tweak, and Scenario parameter definitions are specified in the package *org.drip.param.definition*. It contains the classes that implement the definitions for all parameters except product feature parameters – quotes, calibration parameters, market parameters, tweak parameters, and the scenario curves.

Functionality in this package is implemented across 10 classes and 5 groups – market parameters group (MarketParams, ComponentMarketParams, BasketMarketParams), quote parameters group (Quote, and ComponentQuote), CalibrationParams, Tweak Parameters group (NodeTweakParams, and CreditNodeTweakParams), and the Scenario Curves Group (RatesScenarioCurve and CreditScenarioGroup).

BasketMarketParams

This interface exposes functionality to set/get the named discount curves, credit curves, quotes for each component underlying the basket.

CalibrationParams

This class holds the calibration type and measure – as well as the workout date to which calibration is to be done.

ComponentMarketParams

This interface exposes functionality to set/get the discount curves, credit curves, treasury curves, EDSF curves, quotes, treasury benchmark set, and fixings for the component.

ComponentQuote

This interface exposes functionality to add/remove a quote for a given measure for a component – one of the quotes is a deemed “market quote”.

CreditNodeTweakParams

This class extends the NodeTweakParams, and is meant to be used for CreditCurve tweaks. It holds the tweak parameter and the measure, and indicates whether the tweak operation generates a flat curve.

CreditScenarioCurve

This interface exposes functionality to generate the credit curve from the calibration instruments. It generates the base curve, flat/tenor bumped up/down curve set, as well as custom tweaked scenario curves.

MarketParams

This interface exposes functionality to set/retrieve/remove the following named market objects.

- Named Curves: Rates and Credit Scenario Curves.
- Named Quotes: Component quotes, Treasury benchmark quotes

- Fixings: For a named index and date.
- Component Market Parameters: Parameters for the specified component and scenario.
- Basket Market Parameters: Parameters for the specified basket and scenario.
- Tenor bumped Market Parameters: Array of rates/hazard/recovery component and basket market parameters.

NodeTweakParams

This class holds the node to be adjusted, the adjustment amount, and the type.

Quote

This interface exposes functionality that holds time marked quote value for a given side.

RatesScenarioCurve

This interface exposes functionality to generate the discount curve from the calibration instruments. It generates the base curve, flat/tenor bumped up/down curve set, as well as custom tweaked scenario curves.

Credit Product: Pricing Parameters

Pricing parameter is implemented in the package org.drip.param.pricer. It contains the pricing parameters corresponding to a given product and model.

Currently only the credit-pricing model is implemented – it is implemented in PricerParams.

PricerParams

This class implements the discretization scheme, and indicates whether the pricing also includes calibration.

Credit Product: Valuation Parameters

Valuation parameters are implemented in the package *org.drip.param.valuation*. It contains all the non-market and non-product parameters needed for valuing a product at a given date.

Functionality in this package is implemented across 4 classes – QuotingParams, CashSettleParams, WorkoutInfo, and ValuationParams.

CashSettleParams

This class is used the settle date from the given valuation date from the lag, the holiday calendar, and the date adjustment mode.

QuotingParams

This class contains the parameters required to interpret a component quote – whether the quote is spread/price, and, if it is yield quoted, the quote day count, frequency, calendar, and the quote parameters.

ValuationParams

This class contains the valuation date, the cash settle date, and the holiday calendar.

WorkoutInfo

This class contains the workout information details – the work-out date, yield, the work-out exercise factor, as well as the work-out type.

Credit Product: Product Definitions

Product definitions are implemented in the package *org.drip.product.definition*. It contains interface definitions for all products, along with definitions for credit, rates, and FX components and specific credit/rates/FX products, and baskets.

Product definitions are implemented in different groups – base component group (Component, ComponentMarketParamsRef, CalibrateComponent), base basket group (BasketMarketParamRef, BasketProduct), RatesComponent, Credit Component Group (CreditComponent, CreditDefaultSwap, BondProduct, Bond), and FX Component group (FXSpot and FXForward).

BasketMarketParamRef

This interface exposes the functionality to get the array of IR and credit curves relevant to the basket.

BasketProduct

This abstract class extends BasketMarketParamRef. It provides methods for accessing the basket's components, coupon, notional, effective date, maturity date, cash amount, and the list of coupon periods.

Bond

This abstract class extends CreditComponent. Apart from exposing the a bond specific customized versions of the Component interface, the following are some of the more specialized functionality it provides:

- Analytics functionality for a set of <<FROM>> and <<TO>> measures, e.g., Price from Yield, where Price is the <<TO>> measure and Yield is the <<FROM>> measure. For a full set of measures exposed, please consult CreditAnalytics documentation.
- Spread computed from a set of built-in treasury benchmarks
- ISIN, CUSIP, Ticker, Name
- Period a given date corresponds to
- Price from bumped discount/zero/ credit curves
- Full set of RV measures for a given market measure

For information on the more targeted functionality, please consult the CreditAnalytics documentation.

BondProduct

This interface provides the functionality to get/set the following product features of the bond: bond identifier, credit/rates parameters, treasury benchmark, coupon parameters, (optional) cash flow periods, notional schedules, currency parameters, floater parameters, market quote convention, termination events, and embedded call/put schedules.

Component

This abstract class extends the ComponentMarketParamRef interface, and provides methods for accessing the component's coupon, notional, effective date, maturity date, cash amount, and the list of coupon periods.

ComponentCalibrator

This abstract class extends Component, and provides the calibration functionality and the component's primary/secondary market code.

ComponentMarketParamRef

This interface exposes the functionality to get the curves relevant to valuing the basket – the rates/credit/treasury/EDSF curves.

CreditComponent

This abstract class extends the CalibratableComponent – and provides the functionality needed of the credit component. It retrieves the component's recovery, credit settings, and the coupon/loss period flows.

CreditDefaultSwap

This abstract class extends the CreditComponent, and provides functionality to calibrate a flat spread, as well as to reset the coupon.

FXForward

This abstract class implements the FX forward contract, and provides functionality to retrieve the currency pair, date, product codes, effective/maturity dates, and to imply the forward value and basis, as well as compute a full set of measures.

FXSpot

This abstract class implements the FX spot contract, and provides functionality to retrieve the currency pair and the spot date.

RatesComponent

This hollow abstract class extends the CalibratableComponent – and exists only to indicate the RatesComponent type.

Credit Product: Product Parameters

Product parameter definitions are implemented in the package org.drip.product.params. It contains the implementations of the features required for a complete construction of an instance of the product.

Product parameters are implemented across 19 classes. Validatable is the base interface that underpins most of them. Others are identifier parameters (CDXIdentifier, CDXRefDataParams, IdentifierSet, StandardCDXParams), CouponSetting, CreditSetting, CurrencySet, EmbeddedOptionSchedule, FactorSchedule, NotionalSchedule, PeriodGenerator, PeriodSet, FloaterSetting, RatesSeting, TerminationSetting, QuoteConvention, Treasury Parameters (TreasuryBenchmark, TsyBmkSet), and CurrencyPair.

CDXIdentifier

CDXIdentifier contains the set of fields to uniquely identify a CDX – the index, series, version, and the tenor.

CDXRefDataParams

CDXRefDataParams contains the full set of fields needed to construct the reference data corresponding to a CDX.

CouponSetting

CouponSetting contains the set of coupon parameters. It holds the type, the nominal coupon, the coupon ceiling/floor, as well as the coupon schedule.

CreditSetting

CreditSetting contains the component specific credit parameters – the credit curve name, recovery, recovery pay lag, and a flag indicating whether accrual is paid on default.

CurrencyPair

CurrencyPair contains the currency set details of the FX contact – the numerator, the denominator, and the quoted currencies, as well as the contract PIP factor.

CurrencySet

CurrencySet contains the component's currency fields – the trade, coupon, and redemption currencies.

EmbeddedOptionSchedule

EmbeddedOptionSchedule contains the component's embedded option parameters – the schedule, exercise factors, type, call notice period, and additional parameters if the bond is fixed-to-float on exercise

FactorSchedule

FactorSchedule simply contains an array of dates and doubles – it is used to represent such objects as coupon or notional schedules.

FloaterSetting

FloaterSetting contains the component's floating rate parameters – the rate index, floater day count, spread, and the current coupon.

IdentifierSet

IdentifierSet contains the component's identifier parameters – the generic ID, ISIN, CUSIP, and the ticker.

NotionalSetting

NotionalSetting contains the component's notional schedule parameters – the notional schedule, the notional factor interpretation mode, and whether the price quote is off of the original notional.

PeriodGenerator

PeriodGenerator extends PeriodSet. It contains the date adjustment parameters corresponding to each of the period's dates, as well as other period generation details.

PeriodSet

PeriodSet holds the component's period generation parameters, as well as the generated period set. It holds the date adjustment parameters for the period start/end, period accrual start/end, effective/maturity/pay/reset, as well as first coupon and interest accrual start dates.

QuoteConvention

QuoteConvention contains the component's market convention parameters – the quoting and the cash settle parameters, the calculation type, and the first settle date.

RatesSetting

RatesSetting holds the names of the component's rates curve for each of the cash flow currencies – the trade, the coupon, the principal, and the redemption currencies.

StandardCDXParams

StandardCDXParams contains the parameters that are used for constructing a CDX family – currency, coupon, and the number of components.

TerminationSetting

TerminationSetting indicates the termination mode of a component – whether it has been exercised, if it has defaulted, or if it is perpetual.

TreasuryBenchmark

TreasuryBenchmark holds the component's treasury benchmark and curves. It consists of the TsyBmkSet instance as well as the treasury and EDSF discount curves.

TsyBmkSet

TsyBmkSet holds the treasury benchmarks corresponding to the component – the primary, and an array of the secondary treasury benchmarks.

Validatable

The Validatable interface is implemented by most of the product parameters – it is used to validate the state of the constructed parameter object.

Credit Product: Product RV and Batch Calculation Outputs

Product bulk outputs are implemented in the package *org.drip.analytics.output*. It contains the bulk results of pricing and relative value calculation for the products.

Outputs are implemented in 6 classes – ComponentMeasures, bond specific calculation outputs (ExerciseInfo, BondCouponMeasures, BondWorkoutMeasures, BondRVMeasures), and BasketMeasures.

BasketMeasures

This class contains the set of basket run outputs for a full sequence scenarios – and holds the following:

- Base measures
- Flat credit/rates/recovery bumped delta and gamma measure set
- Tenor bumped credit/rates/recovery bumped delta and gamma measure set
- Flat credit/rates/recovery bumped delta and gamma measure set for each market credit and rates curve bump
- Tenor bumped credit/rates/recovery bumped delta and gamma measure set for each market credit and rates curve bump

BondCouponMeasures

BondCouponMeasures contains the bond's coupon related PV measures – the DV01, the coupon/index/full PV, and the derived measures (i.e., clean/dirty etc) thereof.

BondRVMeasures

BondRVMeasures contains the bond's full set of implied relative value measures – the price, yield, basis of different types (bond basis, credit basis, discount margin), spread measures relative to different base curves (Z, asset swap, treasury, G/I/OAS, PECS), duration, and convexity.

BondWorkoutMeasures

BondWorkoutMeasures contains the bond's full set of measures to a specified workout date – e.g., the clean/dirty, credit risky/risk-less variants of the coupon measures, par, principal, and recovery PV. For a full set of measures please consult the CreditAnalytics documentation.

ComponentMeasures

This class contains the set of component run outputs for a full sequence scenarios – and holds the following:

- Base measures, the flat credit/rates/recovery bumped delta and gamma measure set
- Tenor bumped credit/rates/recovery bumped delta and gamma measure set

ExerciseInfo

ExerciseInfo contains the set of fields that fully characterize and exercise event – the exercise type, date, and the factor.

Credit Product: Serializer

Serializer interface are implemented in the package *org.drip.service.stream*. The interface defines methods for serializing out of and de-serializing into a byte stream, as well as the object serialization version.

There is just one interface in this package – Serializer.

Serializer

The serializer exposes methods for packing and unpacking the implementing object onto a byte array. It also holds the version, which shows the implemented version instance.

Credit Analytics Library

Credit Analytics Library consists of the following 16 packages:

1. Curve Implementations: This contains the curve objects implemented using one of the many ways of calibration, and contains concrete implementations of Discount Curve, Zero Curve, Credit Curve, and FX Curves.

2. Curve Calibrators: This contains the curve calibrators that use different calibration schemes, and the curve scenario generators.

3. Curve Creators: This contains the curve object factories for the different curves using the calibration parameters.

4. Reference Data Loaders: This package contains functionality that loads the bond and the CDS reference data, as well as closing marks for a few date ranges.

5. Analytics Configurator: This package contains functionality to configure various aspects of Credit Analytics.

6. Market, Quote, and Scenario Parameter Implementations: This contains the implementations of the Credit Product interfaces representing the quotes, the basket/component market parameters, and the scenario curve containers.

7. Market, Quote, and Scenario Parameter Creators: This contains the builder factories for the quotes, market parameters, and the scenario curves.

8. Rates Component Implementations: This contains the implementations of the Credit Product interfaces for Cash, Euro-dollar future, and interest rate swap instruments.

9. Credit Product Implementations: This contains the implementations of the Credit Product interfaces for Bonds, CDS, basket CDS, and bond baskets.

10. FX Product Implementations: This contains the implementation of the Credit Product interface for FX products.

11. Product Creators: This contains the creators for the various rates, credit, and FX component and basket products.

12. Analytics Environment Manager: This provides functionality for loading products from their reference data and managing them, as well as creating/accessing live/closing curves.

13. Analytics Bridge: This provides the stub and proxy functionality for invoking Credit Analytics functionality in a remote server and extracting the results.

14. Analytics API: This provides a unified and comprehensive functional, static interface of all the main Credit Analytics functionality.

15. Analytics Samples: This provides samples illustrating the functionality provided by Credit Analytics – samples demonstrating the creation and usage of curves and products across rates, credit, and FX components and baskets. Examples are also provided on how to compare against standard analytics vendors/suppliers (e.g., Bloomberg).

16. Functional Testers: This contains a fairly extensive set of unit and composite testers for the curve, products, serialization, and analytics functionality provided by the Credit Analytics suite, with a special focus on bonds.

Credit Analytics: Curve Implementations

Credit Product curve definitions are implemented in the package *org.drip.analytics.curve*. This package contains the curve objects implemented using one of the many ways of calibration, and contains concrete implementations of Discount Curve, Zero Curve, Credit Curve, and FX Curves.

Functionality in this package is implemented over 5 classes – CalibratedDiscountCurve, DerivedZeroCurve, CalibratedCreditCurve, DerivedFXBasisCurve, and DerivedFXForwardCurve.

CalibratedCreditCurve

This class extends Credit Product's CreditCurve interface. It maintains the term structure of hazard nodes and recovery rates, and the calibration instruments, quotes, measures, and other parameters.

CalibratedDiscountCurve

This class extends Credit Product's DiscountCurve interface. It maintains the term structure of forward rates, and the calibration instruments, quotes, measures, and other parameters.

DerivedFXBasisCurve

This class extends Credit Product's FXBasisCurve interface. It maintains the term structure of the FX Basis nodes either as FX Forward points, or as bootstrapped nodes.

DerivedFXForwardCurve

This class extends Credit Product's FXForwardCurve interface. It maintains the FX Spot instance and the term structure of FX Forwards either as outrights or PIPs.

DerivedZeroCurve

This class extends Credit Product's ZeroCurve interface. It maintains the term structure of spot zeroes for the specified input pay nodes.

Credit Analytics: Curve Calibrators

Curve calibrators are implemented in the package *org.drip.analytics.calibration*. This package contains the curve calibrators that use different calibration schemes, and the curve scenario generators.

Functionality in this package is implemented over 5 classes – ComponentCalibrator, BracketingCalibrator, NewtonRaphsonCalibrator, IRCurveScenarioGenerator, and CreditCurveScenarioGenerator.

BracketingCalibrator

This class implements the ComponentCalibrator interface. Calibration is done through bracketing.

ComponentCalibrator

ComponentCalibrator interface defines the component curve calibration methods – boot-strapping the discount and the hazard curves form the individual component quotes. Calibration is done node-by-node, or flat.

CreditCurveScenarioGenerator

CreditCurveScenarioGenerator holds the calibration and parameters to be used in conjunction with the ComponentCalibrator to generate scenario credit curves.

NewtonRaphsonCalibrator

This class implements the ComponentCalibrator interface. Calibration is done by Newton Raphson technique.

RatesCurveScenarioGenerator

RatesCurveScenarioGenerator holds the calibration and parameters to be used in conjunction with the ComponentCalibrator to generate scenario rates curves.

Credit Analytics: Curve Creators

Curve creators are implemented in the package *org.drip.analytics.creator*. This contains the curve object factories for the different curves using the calibration parameters.

Functionality in this package is implemented over 5 classes – DiscountCurveBuilder, ZeroCurveBuilder, CreditCurveBuilder, FXBasisCurveBuilder, and FXForwardCurveBuilder.

CreditCurveBuilder

CreditCurveBuilder provides several ways of constructing an instance of the Credit Product's CreditCurve interface.

DiscountCurveBuilder

DiscountCurveBuilder provides several ways of constructing an instance of the Credit Product's DiscountCurve interface.

FXBasisCurveBuilder

FXBasisCurveBuilder provides several ways of constructing an instance of the Credit Product's FXBasisCurve interface.

FXForwardCurveBuilder

FXForwardCurveBuilder provides several ways of constructing an instance of the Credit Product's FXForwardCurve interface.

ZeroCurveBuilder

ZeroCurveBuilder provides several ways of constructing an instance of the Credit Product's ZeroCurve interface.

Credit Analytics: Reference Data Loaders

Data loaders are implemented in the package *org.drip.feed.loaders*. This package contains functionality that loads the bond and the CDS reference data, as well as closing marks for a few date ranges.

Functionality in this package is implemented over 3 classes – BondRefData, CDXRefData, and CreditStaticAndMarks.

BondRefData

BondRefData uploads the specified set of bond reference data from an externally supplied reference data file.

CDXRefData

CDXRefData uploads the specified set of the standard CDX definitions from an externally specified file.

CreditStaticAndMarks

CreditStaticAndMarks populates an internal test CreditAnalytics database with bond and CDS reference data, as well as some marks.

Credit Analytics: Analytics Configurator

Credit Analytics configurator is implemented in the package *org.drip.param.config*. This package contains functionality to configure various aspects of Credit Analytics.

Functionality in this package is implemented in a single class – ConfigLoader.

ConfigLoader

ConfigLoader loads the set of CreditAnalytics configuration settings. Please refer to the CreditAnalytics documentation for details on the configuration entries.

Credit Analytics: Market Parameters, Quotes, and Scenario Parameter Implementations

Quotes and Market Parameters are implemented in the package org.drip.param.market. This contains the implementations of the Credit Product interfaces representing the quotes, the basket/component market parameters, and the scenario curve containers.

Functionality in this package is implemented over 7 classes – MultiSidedQuote, ComponentMultiMeasureQuote, BasketMarketParamSet, ComponentMarketParamsSet, MarketParamsContainer, RatesCurveScenarioContainer, and CreditCurveScenarioContainer.

BasketMarketParamSet

BasketMarketParamSet provides the concrete implementation of Credit Product's BasketMarketParams interface. It holds the named discount/credit curves, the named treasury and component quotes, and the fixings object.

ComponentMarketParamSet

ComponentMarketParamSet provides the concrete implementation of Credit Product's ComponentMarketParams interface.

ComponentMultiMeasureQuote

ComponentMultiMeasureQuote provides the concrete implementation of Credit Product's ComponentQuote interface.

CreditCurveScenarioContainer

CreditCurveScenarioContainer provides the concrete implementation of Credit Product's CreditScenarioCurve interface. It holds the credit scenario generator object that contains the calibration instrument/parameters, base credit curve, parallel bump/down credit curve, tenor bumped up/down credit curves, as well as named custom credit curves.

MarketParamsContainer

MarketParamsContainer provides the concrete implementation of Credit Product's MarketParams interface. It holds the full set of market parameters for the given day – named component/treasury quote map, discount/credit curve map, credit/rates scenario curve map, and the fixings map.

MultiSidedQuote

MultiSidedQuote provides the concrete implementation of Credit Product's Quote interface.

RatesCurveScenarioContainer

RatesCurveScenarioContainer provides the concrete implementation of Credit Product's RatesScenarioCurve interface. It holds the rates scenario generator object that contains

the calibration instrument/parameters, base rates curve, parallel bump/down rates curve, tenor bumped up/down rates curves, as well as named custom rates curves.

Credit Analytics: Market Parameters, Quotes, and Scenario Parameter Creators

Builders for quotes, market parameters, and scenario curves are implemented in the package *org.drip.param.creator*. This contains the builder factories for the quotes, market parameters, and the scenario curves.

Functionality in this package is implemented over 7 classes – QuoteBuilder, ComponentQuoteBuilder, ComponentMarketParamsBuilder, BasketMarketParamsBuilder, MarketParamsBuilder, RatesScenarioCurveBuilder, and CreditScenarioCurveBuilder.

BasketMarketParamsBuilder

BasketMarketParamsBuilder provides multiple ways of constructing an instance of the Credit Product's BasketMarketParams interface.

ComponentMarketParamsBuilder

ComponentMarketParamsBuilder provides multiple ways of constructing an instance of the Credit Product's ComponentMarketParams interface.

ComponentQuoteBuilder

ComponentQuoteBuilder provides multiple ways of constructing an instance of the Credit Product's ComponentQuote interface.

CreditScenarioCurveBuilder

CreditScenarioCurveBuilder provides several ways of constructing an instance of the Credit Product's CreditScenarioCurve interface.

MarketParamsBuilder

MarketParamsBuilder provides several ways of constructing an instance of the Credit Product's MarketParams interface.

QuoteBuilder

QuoteBuilder provides several ways of constructing an instance of the Credit Product's Quote interface.

RatesScenarioCurveBuilder

RatesScenarioCurveBuilder provides several ways of constructing an instance of the Credit Product's RatesScenarioCurve interface.

Credit Analytics: Rates Component Implementations

Rates components are implemented in the package *org.drip.product.rates*. This contains the implementations of the Credit Product interfaces for Cash, Euro-dollar future, and interest rate swap instruments.

Functionality in this package is implemented over 3 classes – CashComponent, EDFComponent, and IRSComponent.

CashComponent

CashComponent implements the Credit Product's RatesComponent interface for the Cash product.

EDFComponent

EDFComponent implements the Credit Product's RatesComponent interface for the Euro-Dollar Future product.

IRSComponent

IRSComponent implements the Credit Product's RatesComponent interface for the IRS product.

Credit Analytics: Credit Product Implementations

Credit product definitions are implemented in the package *org.drip.product.credit*. This contains the implementations of the Credit Product interfaces for Bonds, CDS, basket default swaps, and bond baskets.

Functionality in this package is implemented over 4 classes – BondComponent, BondBasket, CDSComponent, and CDSBasket.

BondComponent

BondComponent implements the Credit Product's Bond and BondProduct interfaces for the Bond product.

BondBasket

BondBasket implements the Credit Product's BasketProduct interface for the Bond Basket product. It holds the basket name, basket notional, component bonds, and their weights.

CDSComponent

CDSComponent implements the Credit Product's CreditDefaultSwap interface for the CDS product.

CDSBasket

CDSBasket implements the Credit Product's BasketProduct interfaces for the CDS Basket product. It holds the basket name, basket notional, component CDS'es, and their weights.

Credit Analytics: FX Component Implementations

FX components are implemented in the package *org.drip.product.fx*. This contains the implementations of the Credit Product interfaces for FX spot and forward contracts.

Functionality in this package is implemented over 2 classes – FXSpotContract and FXForwardContract.

FXForwardContract

FXForwardContract represents the FX Forward contact – it contains the spot/maturity dates, the component code, and the CurrencyPair.

FXSpotContract

FXSpotContract represents the FX Spot instance – it contains the spot date and the CurrencyPair.

Credit Analytics: Product Creators

Product creators are implemented in the package *org.drip.product.creator*. This contains the creators for the various rates, credit, and FX component and basket products.

Functionality in this package is implemented over 14 classes – CashBuilder, EDFutureBuilder, IRSBuilder, CDSBuilder, bond creator classes (BondRefDataBuilder, BondProductBuilder, BondBuilder), CDS basket creator classes (CDSBasketBuilder, CDXRefDataBuilder, CDXRefDataHolder, StandardCDXManager), BondBasketBuilder, and FX product builder classes (FXSpotBuilder and FXForwardBuilder).

BondBasketBuilder

BondBasketBuilder provides several ways of constructing an instance of the Credit Product's BasketProduct interface for the Bond Basket.

BondBuilder

BondBuilder provides several ways of constructing an instance of the Credit Product's Bond interface for the BondComponent.

BondProductBuilder

BondProductBuilder contains the set of static parameters needed to construct the bond for the full bond valuation. Please refer to the CreditAnalytics documentation for the full set of bond product valuation fields.

BondRefDataBuilder

BondRefDataBuilder contains the set of bond reference data fields, most of which are not used for valuation. Please refer to the CreditAnalytics documentation for the full set of fields.

CashBuilder

CashBuilder contains the functionality to create a Cash Product instance from different types of inputs. The Credit Product rates product interface RatesComponent is the return type.

CDSBasketBuilder

CDSBasketBuilder contains the functionality to create a CDS Basket Product instance from different types of inputs. An instance of the Credit Product interface BasketProduct is returned.

CDSBuilder

CDSBuilder contains the functionality to create a CDS Product instance from different types of inputs. An instance of the Credit Product interface CreditDefaultSwap is returned.

CDXRefDataHolder

CDXRefDataHolder contains all the generated standard CDX Products, returned as instances of CreditProduct's BasketProduct interface. Since this is a generated file, please do not delete this.

EDFutureBuilder

EDFutureBuilder contains the functionality to create a Euro-dollar Product instance from different types of inputs. The Credit Product rates product interface RatesComponent is the return type.

FXForwardBuilder

FXForwardBuilder contains the functionality to create a FX forward contract instance from different types of inputs. The Credit Product rates product interface FXForward is the return type.

FXSpotBuilder

FXSpotBuilder contains the functionality to create an FX spot contract instance from different types of inputs. The Credit Product rates product interface FXSpot is the return type.

IRSBuilder

IRSBuilder contains the functionality to create an IRS Product instance from different types of inputs. The Credit Product rates product interface RatesComponent is the return type.

Credit Analytics: Analytics Environment Manager

Analytics Environment Manager component are implemented in the package *org.drip.service.env*. This contains the creators for the various rates, credit, and FX component and basket products.

Functionality in this package is implemented over 7 classes – BondManager, CDSManager, EnvManager, EODCurves, RatesManager, StandardCDXManager, and StaticBACurves.

BondManager

BondManager holds the live/closing marks as well as the valuation/ref data for a given bond.

CDSManager

CDSManager holds the live/closing CDS marks as well as the valuation/ref parameters for a given CDS. It also holds the calibrated live and closing credit curves.

EnvManager

EnvManager sets the environment and connection parameters, and populates the market parameters (quotes, curves, and fixings) for a given EOD.

EODCurves

EODCurves implements the container the exposes the functionality to create/retrieve the set of closing rates/credit curves for a given EOD.

RatesManager

RatesManager manages the creation/loading of the rates curves of different kinds for a given EOD.

StaticBACurves

StaticBACurves implements the class that creates a set of discount and credit curves from a user defined set of marks for a given EOD.

StandardCDXManager

StandardCDXManager retrieves the CDX Product, and the static details of all the NA, EU, SovX, EMEA, and ASIA standardized CDS indices. The indices are returned as instances of CreditProduct's BasketProduct interface.

Credit Analytics: Analytics Bridge

Analytics Bridge is implemented in the package *org.drip.service.bridge*. This provides the stub and proxy functionality for invoking Credit Analytics functionality in a remote server and extracting the results.

Functionality in this package is implemented over 2 classes – CreditAnalyticsStub and CreditAnalyticsProxy.

CreditAnalyticsProxy

This class implements the Credit Analytics client thunking proxy. It captures the requests for the Credit Analytics Server from the caller, formats them, and marshals them to the server, and unmarshals the results back to the caller.

CreditAnalyticsStub

This class implements the Credit Analytics server stub. It receives the requests from the client, de-serializes it and invokes the CreditAnalytics functionality, extracts the results, and serializes and sends them back to the client.

Credit Analytics: Analytics API

Analytics API is implemented in the package *org.drip.service.api*. This provides a unified and comprehensive functional, static interface of all the main Credit Analytics functionality.

Functionality in this package is implemented over a single class – CreditAnalytics.

CreditAnalytics

CreditAnalytics is the main functional API interface to all the Credit Analytics functionality. The functionality ranges from date generation and day count to curve/product building and valuation – please refer to the CreditAnalytics documentation for details.

Credit Analytics: Samples

Credit Analytics samples are available in the package *org.drip.service.sample*. This provides samples illustrating the functionality provided by Credit Analytics – samples demonstrating the creation and usage of curves and products across rates, credit, and FX components and baskets. Examples are also provided on how to compare against standard analytics vendors/suppliers (e.g., Bloomberg).

Functionality in this package is implemented over 12 classes – BloombergCDSW, BondAnalyticsAPI, BondBasketAPI, BondStaticAPI, BondLiveAndEODAPI, BondStaticAPI, CDSBasketAPI, CDSLiveAndEODAPI, CreditAnalyticsAPI, DayCountAndCalendarAPI, FXAPI, RatesAnalyticsAPI, and RatesLiveAndEODAPI.

BloombergCDSW

This sample demonstrates the reproduction of the calculations in the Bloomberg CDSW screen.

BondAnalyticsAPI

This sample demonstrates the creation of bonds and usage of the bond analytics API.

BondBasketAPI

This sample demonstrates the creation of the bond basket and the usage of the bond basket analytics API.

BondLiveAndEODAPI

This sample demonstrates the extraction of the live/EOD bond quotes and their eventual measures calculation.

BondStaticAPI

This sample demonstrates the extraction of the static parameters for a given bond.

CDSBasketAPI

This sample demonstrates the creation of CDS basket and usage of the CDS basket analytics API.

CDSLiveAndEODAPI

This sample demonstrates the extraction of the live/EOD CDS quotes and their eventual measures calculation.

CreditAnalyticsAPI

This sample demonstrates the creation of CDS, credit curve creation, and usage of the CDS/credit analytics API.

DayCountAndCalendarAPI

This sample demonstrates the usage of the day-count and the calendar API.

FXAPI

This sample demonstrates the creation of FX Spot/Forward contracts, FX basis/forward curve creation, and usage of the FX analytics/valuation API.

RatesAnalyticsAPI

This sample demonstrates the creation of rates products, IR curve creation, and usage of the rates product valuation and analytics.

RatesLiveAndEODAPI

This sample demonstrates the extraction of the live/EOD rates product quotes and their eventual measures calculation.

Credit Analytics: Functional Testers

Credit Analytics functional testers are available in the package *org.drip.tester.functional*. This contains a fairly extensive set of unit and composite testers for the curve, products, serialization, and analytics functionality provided by the Credit Analytics suite, with a special focus on bonds.

Functionality in this package is implemented over 4 classes – BondTestSuite, CreditAnalyticsTestSuite, ProductTestSuite, and SerializerTestSuite.

BondTestSuite

BondTestSuite tests the bond's valuation using the EOD market curves, the bond reference data, and the EOD bond marks.

CreditAnalyticsTestSuite

CreditAnalyticsTestSuite does a complete test of all the APIs exposed in the CreditAnalytics functional interface.

ProductTestSuite

ProductTestSuite performs a comprehensive test of the key CreditAnalytics product/curve/parameter functionality:

- Products tested: Cash, EDF, IRS, Bonds, CDS, bond basket, and CDS basket.

- Curves tested: Credit and discount curve generation and elaborate bumped scenario metric reconciliation
- Scenarios tested: Base, flat/tenor bumped up/down rates/credit/recovery/treasury quotes adjusted curve set.
- Measures tested: Brief, modest, or fully comprehensive set of measure generation tests.

SerializerTestSuite

SerializerTestSuite serializes, de-serializes, and reconciles every serializable class in the Credit Product and Credit Analytics library suite.

Regression Suite Library

Regression Suite Library consists of the following 2 packages:

1. Core Regression Library: This contains the full set of Regression Suite's core framework and the set of extensible interfaces.

2. Regression Sample: This contains the samples for regression testing of the Credit Analytics library – the samples illustrate creation of the corresponding regression engines and the curve regressors.

Regression Suite: Core

The core functionality of the regression suite library is implemented in the package *org.drip.regression.core*. This contains the full set of Regression Suite's core framework and the set of extensible interfaces.

Functionality in this package is implemented over 8 classes – RegressionEngine, RegressionRunDetail, RegressionRunOutput, RegressionUtil, RegressorSet, UnitRegressionExecutor, UnitRegressionStat, and UnitRegressor.

RegressionEngine

RegressionEngine provides the control and the framework functionality for the entire RegressionSuite. In particular, it provides:

- Control on the level of the regression output detail – module level, aggregated module level, or full roll up.
- Regression setup control – the number of runs, execution environment tuning
- System control parameters – CPU execution times, memory usage, IO/system wait times on a unit regressor granularity
- Statistics generator control

RegressionRunDetail

RegressionRunDetail contains the named field level details of the output of the regression activity.

RegressionRunOutput

RegressionRunOutput holds the detail of a single regression run – regression scenario name, start/end instants, the execution times, and the corresponding regression run detail.

RegressionUtil

RegressionUtil implements the collection of utilities required for completion/calculation of the component and functional regression, e.g., tolerance checks.

RegressorSet

RegressorSet interface exposes the regressor set stubs – setting up and retrieving the named regressor sets.

UnitRegressionExecutor

UnitRegressionExecutor - the abstract class that implements the UnitRegressor interface - is extended by the individual regressor. It initializes, sets up, invokes the derived regressor, cleans up, and compiles statistics on the regression run.

UnitRegressionStat

UnitRegressionStat provides statistics of the execution time and other measures.

UnitRegressor

UnitRegressor interface exposes the unit regressor functionality – the “regress” method and the regressor name.

Regression Suite: Sample

The core functionality of the regression suite library is implemented in the package org.drip.regression.sample. This contains the samples for regression testing of the Credit Analytics library – the samples illustrate creation of the corresponding regression engines and the curve regressors.

Functionality in this package is implemented over 5 classes – DiscountCurveRegressor, ZeroCurveRegressor, CreditCurveRegressor, FXCurveRegressor, and CreditAnalyticsRegressionEngine.

CreditAnalyticsRegressionEngine

CreditAnalyticsRegressionEngine provides the sample implementation of the Regression Suite's RegressionEngine interface for the Credit Analytics library.

CreditCurveRegressor

CreditCurveRegressor provides the sample implementation of the Regression Suite's RegressorSet interface for the Credit Product's CreditCurve interface.

DiscountCurveRegressor

DiscountCurveRegressor provides the sample implementation of the Regression Suite's RegressorSet interface for the Credit Product's DiscountCurve interface.

FXCurveRegressor

FXCurveRegressor provides the sample implementation of the Regression Suite's RegressorSet interface for the Credit Product's FXBasisCurve and FXForwardCurve interfaces.

ZeroCurveRegressor

ZeroCurveRegressor provides the sample implementation of the Regression Suite's RegressorSet interface for the Credit Analytics' ZeroCurve interface.

Usage Notes

This section shows API references and sample file pointers for some of the more commonly used functionality, as well as a few typical vendor calculations replication modules.

API References And Sample Files

Function	API	Sample Fil
Holidays for a calendar set	CreditAnalytics.GetHolsInYear Convention.HolidaySet	DayCountAndCale
Day count year fraction	CreditAnalytics.YearFraction Convention.YearFraction	DayCountAndCale
Date Adjustment and Roll	CreditAnalytics.Adjust CreditAnalytics.RollDate	DayCountAndCale
Rule based period generation	CouponPeriod.GeneratePeriodsBackward CouponPeriod.GeneratePeriodsForward	CouponPeri
Cash Product definition and creation	CashBuilder.CreateCash	RatesCompon CashBuilde RatesAnalytics ProductTestS
Euro-dollar Future definition and creation	EDFBuilder.CreateEDF EDFBuilder.GenerateEDFPack	RatesCompon EDFBuilde RatesAnalytics ProductTestS

Interest Rate Swap definition and creation	IRSBuilder.CreateIRS	RatesCompon IRSBuilde RatesAnalytics ProductTestS
CDS definition and creation	CDSBuilder.CreateCDS	CreditDefaultS CDSBuilde CreditAnalytic ProductTestS
Bond definition and creation	BondBuilder.CreateSimpleFixed BondBuilder.CreateSimpleFloater BondBuilder.CreateBondFromCF BondBuilder.CreateBondFromParams	Bond BondProduc BondBuilde BondAnalytics BondTestSu ProductTestS
FX Spot/Forward Contact definition and creation	FXSpotBuilder.CreateFXSpot FXForwardBuilder.CreateFXForward	FXSpot FXForward FXSpotBuild FXForwardBu FXAPI
CDS Basket definition and creation	CDSBasketBuilder.MakeCDX CDSBasketBuilder.MakeBasketDefaultSwap StandardCDXManager.GetOnTheRun StandardCDXManager.GetCDXSeriesMap	BasketProdu CDSBasketBu StandardCDXM CDSBasketA ProductTestS
Bond Basket definition and creation	BondBasketBuilder.CreateBondBasket	BasketProdu BondBasketBu BondBasketA ProductTestS
Curve Calibration	ComponentCalibrator.bootstrapHazardRate ComponentCalibrator.bootstrapInterestRate	BondComponent.Bon CDSComponent.Sprea
Discount	DiscountCurveBuilder.CreateDC	RatesAnalytics

Curve definition and creation	RatesScenarioCurveBuilder.CreateDiscountCurve	ProductTestS
Zero Curve definition and creation	ZeroCurveBuilder.CreateZeroCurve	BondCompon
Credit Curve definition and creation	CreditCurveBuilder.CreateCreditCurve CreditScenarioCurveBuilder.CreateCreditCurve	CreditAnalytic ProductTestS
FXBasis Curve definition and creation	FXBasisCurveBuilder.CreateFXBasisCurve	FXAPI
FXForward Curve definition and creation	FXForwardCurveBuilder.CreateFXForwardCurve	FXAPI
Discount Curve Scenario Generation	RatesScenarioCurve.cookScenarioDC RatesScenarioCurve.cookCustomDC	RatesAnalytics ProductTestS
Credit Curve Scenario Generation	CreditScenarioCurve.cookScenarioCC CreditScenarioCurve.cookCustomCC	CreditAnalytic ProductTestS
Discount Curve Usage	DiscountCurve.createRateShiftedParallelCurve DiscountCurve.createBasisRateShiftedCurve DiscountCurve.getDF DiscountCurve.calcImpliedRate	RatesAnalytics ProductTestS
Credit Curve Usage	CreditCurve.createParallelHazardShiftedCurve CreditCurve.createFlatCurve CreditCurve.getSurvival CreditCurve.calcHazard	CreditAnalytic ProductTestS

	CreditCurve.getRecovery	
FX Curve Usage	FXBasisCurve.getFullFXFwd FXForwardCurve.getFullBasis FXForwardCurve.bootstrapBasis FXForwardCurve.bootstrapBasisDC	FXAPI
Component Market Parameters definition and creation	ComponentMarketParamsBuilder.CreateComponentMarketParams	RatesAnalytics BondAnalytics CDSAnalytics ProductTestS
Basket Market Parameters definition and creation	BasketMarketParamsBuilder.CreateBasketMarketParams	BondBasketA CDSBasketA ProductTestS
Cash Product pricing and risk	CashComponent.value	RatesAnalytics ProductTestS
Euro-dollar Future pricing and risk	EDFComponent.value	RatesAnalytics ProductTestS
Interest Rate Swap pricing and risk	IRSComponent.value	RatesAnalytics ProductTestS
CDS pricing and risk	CDSComponent.value	CreditAnalytic ProductTestS
Bond pricing and risk	BondComponent.value	BondAnalytics ProductTestS
Bond RV Analytical	Bond.calc<<TO>>From<<FROM>> e.g., Bond.calcPriceFromYield	BondAnalytics

measures calculation	Bond.standardMeasures CreditAnalytics.Bond<<TO>>From<<FROM>>	
FX Forward Contact pricing and risk	FXForward.value FXForward.implyFXForward FXForward.calcDCBasis	FXAPI
CDS Basket pricing and risk	CDSBasket.value	CDSBasketA ProductTestS
Bond Basket pricing and risk	BondBasket.value	BondBasketA ProductTestS
Product cash flow display	Component.getCouponPeriod BasketProduct.getCouponPeriod	RatesAnalytics CreditAnalytic BondAnalytics ProductTestS

Vendor Pricing Functionality Illustration Files

BloombergCDSW: Illustrates the reproduction of the CDSW command of Bloomberg. It takes inputs of the different Bloomberg curve and product fields precisely as in CDSW, builds the curves, and generates the product measures.

Installation and Deployment Notes

Installation is really simple just drop of each of the jars (CreditProduct, CreditAnalytics, and RegressionSuite) in the class-path.

Configuration is done off of the configuration files corresponding to each of the libraries. For most typical set-ups, the standard configuration should suffice. Please consult the configuration documentation on each of the libraries to configure each of the modules.

Because there is no other dependency, deployment should also be straightforward. Use the regression output as a guide for module capacity estimation.