****

**Vontobel**

**Index Calculation And Reconciliation Tool**

**Technical Design Document**

**(Index Engine)**

Version 1.0

01/16/2018

Contents

[1 Introduction 3](#_Toc398102841)

[1.1 Project Objectives 3](#_Toc398102842)

[2 Design Guidelines and Approach 3](#_Toc398102843)

[2.1 Assumptions / Constraints / Standards 3](#_Toc398102844)

[3 Architecture 3](#_Toc398102845)

[3.1 Servers 3](#_Toc398102846)

[3.2 Software 3](#_Toc398102847)

[3.3 Deployment and Configuration 3](#_Toc398102848)

[4 Functional Users of the Application 4](#_Toc398102849)

[5 Use-Cases and Functionality 4](#_Toc398102850)

[6 Database Design 4](#_Toc398102851)

[6.1 Databases 4](#_Toc398102852)

[6.2 Service Accounts or User Pass Through 4](#_Toc398102853)

[6.3 Project Objects 4](#_Toc398102854)

[6.3.1 Tables 4](#_Toc398102855)

[6.3.2 Views 4](#_Toc398102856)

[6.3.3 Stored Procedures 4](#_Toc398102857)

[6.3.4 Functions 4](#_Toc398102858)

[6.3.5 Other Database Objects 5](#_Toc398102859)

[6.3.6 Data Conversions 5](#_Toc398102860)

[6.4 Referenced Objects 5](#_Toc398102861)

[6.4.1 Tables 5](#_Toc398102862)

[6.4.2 Views 5](#_Toc398102863)

[6.4.3 Stored Procedures 5](#_Toc398102864)

[6.4.4 Functions 6](#_Toc398102865)

[6.4.5 Other Database Objects 6](#_Toc398102866)

[6.5 Data Conversions 6](#_Toc398102867)

[7 User Interface Design 6](#_Toc398102868)

[7.1 Listing of pages or general description of design of User Interface 6](#_Toc398102869)

[8 Security 6](#_Toc398102870)

[8.1 Data Security Classification Level 6](#_Toc398102871)

[8.2 Risk Analysis 7](#_Toc398102872)

[9 User Communications 7](#_Toc398102873)

[10 Application Program Interfaces 7](#_Toc398102874)

[10.1 Incoming Data 7](#_Toc398102875)

[10.2 Outgoing Data 7](#_Toc398102876)

[11 Performance 7](#_Toc398102877)

**Revision History**

**[This document will be continuously updated from initial specification through deployment and future maintenance updates]**

|  |  |  |  |
| --- | --- | --- | --- |
| Change Initiated By | Date | Reason for Changes | Version |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Introduction

[A basic paragraph to describe the project; this will most likely be copied/pasted into all introduction sections of all of the project’s documents. It should be high level and describe the business need that this project is fulfilling.]

## Project Objectives

[General purpose and goals of this project]

# Design Guidelines and Approach

[This section describes the principles and strategies to be used as guidelines when designing and implementing the system. Define the project language, framework, and technologies to utilize. Specify where business logic resides; db vs. app]

## Assumptions / Constraints / Standards

[Describe any general design assumptions / constraints / standards related to any of the project’s design. Examples of Assumptions/Contraints/Standards: able to use JavaScript, application must successfully pass a vulnerability scan, application follows standadards for asp.net-MVC, or webforms….]

# Architecture

## Required Files and their Paths :

* 1. **Directory** : Folder / Directory where **processcaller.py** or **processcaller.exe** exists
  2. **Engine** : <Directory>
  3. **\_\_init\_\_.py** : <Directory/Engine>
  4. **corePrRule.py** : <Directory/Engine>
  5. **corePrDefandDataCol.py** : <Directory/Engine>
  6. **plogger.py** : <Directory/Engine>
  7. **coreOptionPricing.py** : <Directory/Engine>
  8. **coreOptionMetrics.py** : <Directory/Engine>
  9. **MasterMapping.csv** : <Directory>
  10. **Input Files for an Index** : <Directory/VTId of the index>

(ex: <Directory/EiUSXXXXXXXDWSPXXXXML>

## MasterMapping.csv : This CSV is maintained for listing all the defined processes, sub-processes, input files required, output files that will be generated from different processes for all the Indices on-boarded. Headers of the CSV :

* 1. **IndexVTId** : VTId (Indentifier) of the Index
  2. **VTClassName** : Python Class where the business layer of the index is written in crPrRule.py (if the VT calculation is onboarded for the index)
  3. **GenerateBankComposition\_pr** : Process name used for creating the Bank Composition (if Bank calculation is on-boarded for the index)
  4. **GenerateBankComposition\_filenames** : Input files required for generating Bank Composition (if Bank calculation is on-boarded for the index)
  5. **GenerateCompositionLevel\_pr** : Process name used for generating the final Index level (if Bank calculation is on-boarded for the index)
  6. **GenerateCompositionLevel\_filenames** : Input files required for generating the Index level (if Bank calculation is on-boarded for the index)
  7. **GenerateVTComposition\_pr** : Process name used for creating the VT Composition (if VT calculation is on-boarded for the index)
  8. **GenerateVTComposition\_filenames** : Input files required for generating VT Composition (if VT calculation is on-boarded for the index)
  9. **GenerateVTCompositionLevel\_pr** : Process name used for generating the final Index level (if VT calculation is on-boarded for the index)
  10. **GenerateVTCompositionLevel\_filenames** : Input files required for generating the Index level (if VT calculation is on-boarded for the index)
  11. **GenerateVolSurface\_pr** : Process name used for generating Implied Volatility surface for Option Metrics Engine
  12. **GenerateVolSurface\_filenames** : Input files required for generating Implied Volatility surface for Option Metrics Engine
  13. **GenerateForwardCurve\_pr** : Process name used for generating Forward Curve for Option Metrics Engine
  14. **GenerateForwardCurve\_filenames** Input files required for generating Forward Curve for Option Metrics Engine
  15. **GenerateOptionPremium\_pr** : Process name used for calculating option Premium in Option Pricing engine
  16. **GenerateOptionPremium\_filenames** : Input files required for calculating option Premium in Option Pricing engine

## processcaller.py : Imports Engine Module (as engine) . Main function (\_\_name\_\_ == "\_\_main\_\_") takes json\_string (C# argument) as the input which contains the following details.

|  |  |
| --- | --- |
| Process | Name of the Process as per the Master Mapping |
| RunDate | Date for which the Index is being Run |
| LastRunDate | Previous working day to the RunDate |
| IndexVTId | VTId of the Index being Run |
| IndexCurrency | Currency of the Index |
| IsIndexOfIndices | True (if the uderlyings are indices of the existing Indices |
| IsStaticConstituent |  |
| FileDirectory | The Directory in which the required input files are present |
| CreatedBy | System / any Human User |
| CreatedDate | The Date when the process is created / Run |
| LastModifiedBy | System / any Human User |
| LastModifiedDate | Last Modified Date by System / User |

* 1. **fn\_ProcessCall :** This is the function in the processcaller.py called by the Main function which takes C# argument (df\_CInput), directory ( directory in which the package is installed) and log (object created using the cl\_plogger class for logging the whole process).
  2. Other than Log detailing in the above function there are two other classes called from the corePrDefandDataCol.py [cl\_IndexDetails, cl\_ProcessDefiner].
  3. Once the Process is defined using cl\_ProcessDefiner function the sub-processes are mapped from the MasterMapping.csv file using cl\_ProcessMapper (class defined in corePrRule.py)
  4. The respective classes / functions of the index are executed from corePrRule.py and the Index Level and Composition are calculated.

## Engine :

This is the main Module of the Python package. It further contains several sub-modules of which two are general and are used for all the Indices in common.

* 1. **corePrDefandDataCol.py** : This module contains two classes which are used by processcaller.py
     1. **cl\_IndexDetails :** This class takes input from the C# argument (df\_CInput). With all the inputs given this class froms a mapping with the processes in MasterMapping.csv file and fetches all the mapping for the specified index using the function (**fn\_OnboardingDetails**). The arguments needed for this class are
        + 1. IndexVTId
          2. IndexCurrency
          3. IsIndexOfIndices
          4. IsStaticConstituent
          5. Directory
          6. log
     2. **cl\_ProcessDefiner:** This class inherits the above class (**cl\_IndexDetails)**. Apart from that this class takes inputs from the C# argument
        + 1. Process
          2. RunDate
          3. LastRunDate

There are two functions/methods defined in this class which are used reading process from the MasterMapping.csv File and also the helper function used to read data from the C\_input files

1. **fn\_ReadProcess** : This function / method is used to match the input process from the C# argument with the process mentioned in the MasterMapping.csv for the respective index.
2. **fn\_ReadFileData** :
   1. This function / method is used to read data from the C\_ input files that are used for the index/composition calculation in corePrRule.py
   2. All the required input files for the index/composition calculation are listed in the MasterMapping.csv. This function checks whether all the required files are present in the directory and if present reads all the files and makes a dictionary, each key being the filename, (**dict\_FileData**) which is used in the corePrRule.py for calculation purposes
   3. **plogger.py** : This module is used to log all the Success / Failure details for a particular process run for an index. This contains a class **cl\_plogger** which further has several functions / methods. Inputs for initiating this class**:**
      * 1. directory : Directory where the log file is stored
        2. filename : name of the log file
        3. datetimeformat : date and time when the log details are written
3. **log\_directory\_check :** checks whether the directory exists for the log file, if not creates one new directory
4. **writing2file :** This function / method takes the str message as input and writes it to the log file along with the traceback error (if present)
5. **processname\_change:** This function / method takes the input processname that is currently in use and mentions in the log file as the header and the log details are mentioned under it.
6. **error\_reporting:** when the process fails due to some reason that is captured as the Exception in the code and written to the log file using this function
7. **process\_success:** If the process is a success then this function logs it as a success under that after the whole process is executed
8. **process\_failure:** while executing a process if an error occurs in between it goes into the exception (except: ) and the process is logged as failed in the log file
9. **log\_message:** If there is any particular message in between any process (like highlighting if a new file is created ) that can be written to the log file using this function / method

Other than these two there are three more python files which are used for different purposes

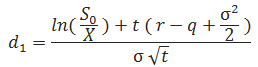
* + 1. **corePrRule.py** : This python file contains the business layer for all the indices which are explained individually below in section 5. There are some general classes defined applicable to all the indices and there are specific classes defined for specific indices.
       1. General Classes:
          1. **PrNames** : Enumeration of Process Names fetched from MasterMapping.csv
          2. **SubPrNames** : Enumeration of Sub-Process Names fetched from MasterMapping.csv
          3. **cl\_ProcessMapper** : Mapping of the above enumerated process and sub-processes with the classes and functions defined in the code.
          4. **cl\_VTCalculation** : This has Mapping for the VT classes names for MasterMapping.csv to the code. There is one method (**fn\_VTOpenCloseComp**) defined under this there are two functions (**fn\_VTCloseComp, fn\_VTOpenComp**) called which are defined in every VT Class.
       2. Specific Classes :
          1. **cl\_BankOCClose** : This class has all the functions/methods for generating close composition for all the bank calculated indices other than Options and Equities
          2. **cl\_BankOCOpen** : This class has all the functions/methods for generating open composition for all the bank calculated indices other than Options and Equities
          3. **cl\_Comp2Level** : This class has functions/methods for generating index levels. This is commom for all the bank and VT calculated indices (including Options and Equities)
          4. **EDEIndexCalculation** : VT class for SEDE Index
          5. **EDUIndexCalculation** : VT class for SEDU Index
          6. **SDIVIndexCalculation** : VT class for SDIV Index
          7. **MRETIndexCalculation** : VT class for MRET Index
          8. **RVOLIndexCalculation** : VT class for RVOL Index
          9. **CUBESDIndexCalculation** : VT class for CUBES-D Excess Return Indices
          10. **CUBESDTRIndexCalculation** : VT class for CUBES-D and Citi Momentum Total Return Indices
          11. **EquityIndexCalculation** : VT class for Style (Equity) Indices
          12. **CitiMomentumIndexCalculation** : VT class for Citi Momentum Excess Return Indices
          13. **cl\_OptionMetricsCalculations** : All the Option Indices whose prices, risk parameters are calculated using IVYDB data are calculated here. DWSP Index underlying prices are calculated. This class uses the **coreOptionMetrics.py** module for calculations. Explained in detail later in Section 5
    2. **coreOptionMetrics.py** : This python file contains code for the Option Metrics calculation (for now Implied Volatility, Forward Curve, Option Premium Calculation)
    3. **coreOptionPricing.py** : This module is imported to coreOptionMetrics.py for Option Premium calculations using different mdels

# Option Metrics Engine

## coreOptionPricing.py

This is a general module written to accommodate various types of Instruments and various pricing models. For now the Instrument handled is Equity Option and the Pricing Model used is BLACKSCHOLESMERTON.

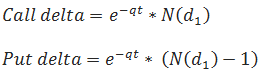
1. **PricingModel** : Enumeration for Pricing Model (BSM
2. **RiskParameter** : Enumeration for calculating Risk Parameters
3. **OptionType** : Enumeration for type of option (Call, Put)
4. **ExpiryType** : Enumeration for type of expiration (European, American)
5. **UndlType** : Enumeration for type of Underlying (Stock, Future)
6. **DivType** : Enumeration for type of Dividend, if any (Discrete, Yield)
7. **Instrument** : This class is used to define an instrument. Three abstract methods are defined under this class. These are to be defined whenever this class is called or inherited.
   1. **payoff**
   2. **price**
   3. **risk\_parameters**
8. **EqOptions** : This class inherits the **Instrument** class. This is specifically defines for Equity Options. It also inherits the class **PricingEngine** where all the selection of the model and the calculation lies.
   1. Other than these various parameters are required for the Equity Option calculator.
      1. **option\_type**
      2. **expiry\_type**
      3. **strike**
      4. **spot0**
      5. **maturity**
      6. **volatility**
      7. **rf\_rate**
      8. **div\_type**
      9. **pv\_div**
      10. **yield\_div**
      11. **model**
   2. Methods / Functions defined under this class
      1. **payoff** : Calculates the Pay-off based on the type of Option
      2. **price** : Calculates the price of the option using the PricingEngine and the specific model used for the type of the option underlying
      3. **risk\_parameters** : Calculates the risk parameters using the PricingEngine
      4. **adj\_spot0** : Spot Price (Security Price) is adjusted based on the type of the Dividend. (If Yield it remains same, if Discrete the PV of the Dividend is subtracted from the Spot Price
      5. **imply\_volatility** : Calculates the Implied Volatility of the Option using the PricingEngine
9. **PricingEngine** : This class is used to map the Pricing Model with the instrument based on its underlying type. Here Equity Options are mapped with BSM model. Methods/Functions defines under this are:
   1. Parameters required for the Pricing Engine
      1. **\_\_mapping\_dict** : Mapping between Pricing Model and type of Instrument
      2. **\_option** : Instrument and its required parameters are defined
   2. Methods / Functions defined under this class
      1. **pricing\_model** : Pricing Model is chosen according to the type of instrument
      2. **valuation** : Price / Premium is calculated for the option using the selected model for the underlying type
      3. **risk\_parameters** : Risk Parameters are calculated for the option using the selected model for the underlying type
      4. **imply\_volatility** : Implied Volatility is calculated for the option using the selected model for the underlying type
10. **Model** : Model is defined. Two abstract methods (**valuation, risk\_parameters**) are defined
11. **BSM** : (BLACKSCHOLESMERTON) This class contains all the calculations for the Option Pricing and Risk Parameter calculations.
    1. Parameters required for the Pricing Engine
       1. **option :** Instrument and its required parameters are defined
    2. Methods / Functions defined under this class
       1. **d1 :**



* + 1. **d2 :**



* + 1. **discount\_factor :** Discounting a value to the present using rf\_rate(Zero Rates)
    2. **div\_discount\_factor** : Discounting of a Dividend to the present using yield\_div (Dividend Yield)
    3. **option\_flag** : Usually in the pricing and risk parameter calculations Call and Put take different signs.
    4. **delta** :



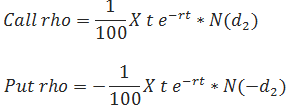
* + 1. **gamma** :



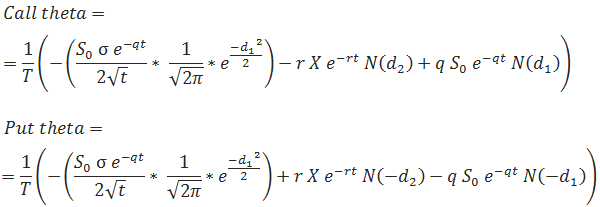
* + 1. **vega** :



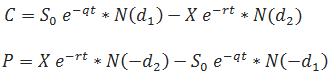
* + 1. **rho** :



* + 1. **theta** :



* + 1. **valuation** :



* + 1. **risk\_parameters** : All the above Methods are called here and made into a dictionary.
    2. **imply\_volatility** : This takes premium as the input parameter. An objective function is defined (valuation formula mentioned above – premium\_input). This function is passed through a optimizer

**optimize.minimize(objective\_func, 0.5,method='L-BFGS-B').x[0]**

## coreOptionMetrics.py

This is the main module for the Option Metrics Engine. Other than general libraries it imports corePrRule and coreOptionPricing modules. This module takes C\_ files as input for Volatility Surface, Forward Curve and Option Pricing calculations and returns P\_ files (output files). This module has only one class (**cl\_OptionMetrics**) defined under which multiple functionalities are provided.

1. Input Files
   * 1. C\_LiborDiscountRateForForwardCurve\_OptionMetrics\_<RunDate>
        1. Required Fields
           1. Date : (Text convertible to Date)
           2. Days : Integer
           3. Rate : Float
        2. This file has data from IVY\_DB data for the zero rates for specific number of days. This data is used and interpolated for required number of days
     2. C\_DividendForForwardCurve\_OptionMetrics\_<RunDate>
        1. Required Fields
           1. InstrumentVTId : Text
           2. SecurityId : Number
           3. Date : (Text convertible to Date)
           4. SecurityPrice : Float
           5. Amount : Float
           6. DistributionType : Text (Identifier)
           7. SourceName : Text
        2. This file fetches data for Discrete dividends and Dividend Yields from IVY\_DB. This data is used to calculate the present value of dividends to adjust the security / spot price
     3. C\_OptionPriceForVolSurface\_OptionMetrics\_<RunDate>
        1. Required Fields :
           1. SecurityId : Number
           2. Date : (Text convertible to Date)
           3. OptionType : Text
           4. Symbol : Text
           5. Strike : Float
           6. MaturityDate : (Text convertible to Date)
           7. ImpliedVolatility : Float
           8. Delta : Float
        2. This file is used for the Volatility surface calculation. A filter is applied on Delta for removing outliers
     4. C\_InputForOptionPremium\_OptionMetrics\_<RunDate>
        1. Required Fields :
           1. UnderlyingVTId : Text
           2. UnderlyingName : Text
           3. OptionType : Text
           4. PricingDate : (Text convertible to Date)
           5. StrikePrice : Float
           6. SpotPrice : Float
           7. MaturityDate : (Text convertible to Date)
           8. RfRate : Float
           9. Volatility : Float
           10. DividendYield : Float
           11. DividendPV : Float
           12. ImpliedVolCalc : Boolean(True/False)
           13. PremiumCalc : Boolean(True/False)
           14. PricingModel : Text
        2. This file is used for the Option Premium or Implied Volatility calculation as per the requirement. This file fetched only when there is a input for web
2. Output Files
   * 1. P\_OptionPricing\_OptionMetrics\_<RunDate>
        1. Required Fields
           1. Delta : Float
           2. Gamma : Float
           3. Price : Float
           4. Rho : Float
           5. Theta : Float
           6. Vega : Float
        2. These are risk parameters calculated and given as output for Web or for the PnL analysis
     2. P\_InputForOptionPremium\_OptionMetrics\_<RunDate>
        1. Required Fields
           1. UnderlyingVTId : Text
           2. UnderlyingName : Text
           3. OptionType : Text
           4. PricingDate : (Text convertible to Date)
           5. StrikePrice : Float
           6. SpotPrice : Float
           7. MaturityDate : (Text convertible to Date)
           8. RfRate : Float
           9. Volatility : Float
           10. DividendYield : Float
           11. DividendPV : Float
           12. ImpliedVolCalc : Boolean(True/False)
           13. PremiumCalc : Boolean(True/False)
           14. PricingModel : Text
        2. This is the output file for Option premium calculations
     3. P\_VolSurface\_OptionMetrics\_<RunDate>
        1. Required Fields
           1. Date : (Text convertible to Date)
           2. Strike : Float
           3. Implied Volatility : Float
        2. This is a json for the Matrix data generated for Implied Volatility Surface
     4. P\_ForwardCurve\_OptionMetrics\_<RunDate>
        1. Required Fields
           1. Date : (Text convertible to Date)
           2. ForwardPrice : Float
        2. This is an output file for the Forward Price calculation for a day in future
3. Input Parameters reuired for the initialization
   * 1. **RunDate** : Run Date for which the calculations are made
     2. **directory** : Directory in which the input and output files are written
     3. **IndexVTId** : VTId of the Index whose underlying (option) prices are calculated
     4. **log** : plogger module imported for logging the process (Success / Failure)
     5. **cl\_FileProcessor** : For reading the C\_ input files
     6. **Fwd\_Days** : Defined to be 726 days (2 years). Can be changed later accordingly
     7. **ZeroRates** : Initiating a Dataframe for Libor Rates calculation (Interpolation)
     8. **matuirty\_days** : (=0) Intiating number of days left for maturity
     9. **temp\_spot** : (=0) Intiating Spot/Security Price
     10. **temp\_vol** : (=0) Initiating initial volatility
     11. **temp\_rfrate** : (=0) Initiating zero rate for the calculations
     12. **temp\_dy** : (=0) Initiating Yield (for Dividend Calculations)
     13. **temp\_dt** : (=0) Initiating type of Dividend
     14. **temp\_dpv** : (=0) Initiating present value of dividend calculations
4. Properties defined for this class
   * 1. **Fwd\_CalDate** : This defines the Future Date (RunDate + Fwd Days)
     2. **IVY\_OPTION\_PRICE** : This generates a dataframe using C\_OptionPriceForVolSurface\_OptionMetrics\_<RunDate> if the processes are any of the genVolSurf, genOptionPremium, genBankComp
     3. **IVY\_DIVIDEND** : This generates a dataframe using C\_DividendForForwardCurve\_OptionMetrics\_<RunDate> if the processes are any of the genFwdCurve, genOptionPremium, genBankComp
     4. **IVY\_LIBOR\_DC :** This generates a dataframe using C\_LiborDiscountRateForForwardCurve\_OptionMetrics\_<RunDate> if the processes are any of the genFwdCurve, genOptionPremium, genBankComp
     5. **SecutiryPrice** : This fetches security price from the **IVY\_DIVIDEND** dataframe if the processes are any of the genFwdCurve, genOptionPremium, genBankComp
     6. **WEB\_OP\_Input** : This generates a dataframe using C\_InputForOptionPremium\_OptionMetrics\_<RunDate> if the processes are any of the genOptionPremium, genBankComp
5. Methods defined for further calculations :
   * 1. **fn\_LnItpVolSurf** : This method uses the dataframe **IVY\_OPTION\_PRICE** for the required data (Strike, Expiry, Volatility) for a particular type of option (Call or Put). This generates a 3D surface using the method (**griddata)** imported from Scipy library for interpolation of IVY\_DB Vol data. Finally the output points, Implied Volatility for a particular strike and maturity are given as output in the file P\_VolSurface\_OptionMetrics\_<RunDate> as a matrix. Also using Matplotlib library a Volatility surface is plotted and sent in PNG foramat P\_VolSurface\_OptionMetrics\_<RunDate>
     2. **fn\_LnItpVolPoint** : This method is used to generate Implied Volatility for a particular Strike and Maturity. The same method (**griddata**) is used for this, moreover two methods (linear or nearest) are used according to the purpose
     3. **fn\_LiborZeroRates** : This method takes input from **IVY\_LIBOR\_DC** dataframe and uses simple **interp1d** from Scipy Module for interpolation. This outputs a dataframe **ZeroRates** which is used in further calculations for forward and oprion price calculations
     4. **fn\_PV\_Dividends** : This inputs data from **IVY\_DIVIDEND** dataframe and fetches all the dividends (both Discrete and Yield). Here only Discrete Dividends are handled. Based on the Distribution type of Dividend the present value of the Dividend is calculated. It is different of different number of days to maturity. This method returs Dividends[["Days", "Dividend"]] dataframe for the forward price calculations
     5. **fn\_DivType** : This method is used to decide if there is any Dividend Yield present in the **IVY\_DIVIDEND** dataframe based on the source of the Dividend
     6. **fn\_FwdCurve** : This method takes input from **fn\_LiborZeroRates, fn\_PV\_Dividends, fn\_DivType.** At first the present value of dividends is adjusted for every calculation date of Forward Price. Then the Forward price is calculated adjusting Dividend Yield if there is any. A final data frame is generated, Forward Price for every calculation date and written to the output file P\_ForwardCurve\_OptionMetrics\_<RunDate>. Also using Matplotlib a 2D plot is generated and sent as a PNG P\_ForwardCurve\_OptionMetrics\_<RunDate>
     7. **fn\_InterimOptionPricing** : This method takes input from **WEB\_OP\_Input** dataframe. If the UnderlyingName is given as Dummy then all the fields are required. If anything misses the output is given as NaN. If the UnderlyingName is one of the on-boarded constituents then we have some data that can be fetched from the database. This method analyses all the parameters required for Option Pricing and sets the values accordingly
     8. **fn\_OptionPricingParameters** : This method makes a dictionary of all the input parameters and passes to the Option Pricing function (EqOptions) imported from coreOptionPricing. Premium or Implied Volatility is calculated according to the requirement. A dataframe with output as Option\_details is created. This method returns two dataframes Option\_details, option\_parameters
     9. **fn\_OptionPricing** : The above function is called here and all the output is written to the file P\_OptionPricing\_OptionMetrics\_<RunDate> and all the option parameters are written to the file P\_InputForOptionPremium\_OptionMetrics\_<RunDate>

# Business Layer - Python

### Index - Introduction

**Name** - Citi Commodities Curve Beta Enhanced Distributed Mono Indices (Citi Commodities CUBES-D Mono Indices)

**Type** – Family of Commodity Indices. Contains 56 Indices (28 based on Excess Return Type and 28 based on Total Return Type)

**List of Indices –**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | ***CUBES Index*** | ***CUBES Index Ticker*** | | ***Exchange*** | ***CCT Ticker*** |
|  | ***(Commodity Contract Type)*** | ***Excess Return Index*** | ***Total Return Index*** |  |  |
| 1 | WTI Crude Oil | CCUDCLER | CCUDCLTR | NYMEX | CL |
| 2 | Brent Crude Oil | CCUDCOER | CCUDCOTR | ICE | CO |
| 3 | Natural Gas | CCUDNGER | CCUDNGTR | NYMEX | NG |
| 4 | Gasoline (RBOB) | CCUDXBER | CCUDXBTR | NYMEX | XB |
| 5 | Heating Oil | CCUDHOER | CCUDHOTR | NYMEX | HO |
| 6 | Gas Oil | CCUDQSER | CCUDQSTR | ICE | QS |
| 7 | Gold | CCUDGCER | CCUDGCTR | COMEX | GC |
| 8 | Silver | CCUDSIER | CCUDSITR | COMEX | SI |
| 9 | Copper (LME) | CCUDLPER | CCUDLPTR | LME | LP |
| 10 | Copper (CMX) | CCUDHGER | CCUDHGTR | COMEX | HG |
| 11 | Aluminum | CCUDLAER | CCUDLATR | LME | LA |
| 12 | Lead | CCUDLLER | CCUDLLTR | LME | LL |
| 13 | Nickel | CCUDLNER | CCUDLNTR | LME | LN |
| 14 | Zinc | CCUDLXER | CCUDLXTR | LME | LX |
| 15 | Tin | CCUDLTER | CCUDLTTR | LME | LT |
| 16 | Wheat (CBOT) | CCUDWXER | CCUDWXTR | CBOT | W |
| 17 | Wheat (KCBOT) | CCUDKWER | CCUDKWTR | KCBOT | KW |
| 18 | Corn | CCUDCXER | CCUDCXTR | CBOT | C |
| 19 | Soybeans | CCUDSXER | CCUDSXTR | CBOT | S |
| 20 | Soybean Oil | CCUDBOER | CCUDBOTR | CBOT | BO |
| 21 | Soybean Meal | CCUDSMER | CCUDSMTR | CBOT | SM |
| 22 | Coffee | CCUDKCER | CCUDKCTR | NYBOT | KC |
| 23 | Sugar | CCUDSBER | CCUDSBTR | NYBOT | SB |
| 24 | Cocoa | CCUDCCER | CCUDCCTR | NYBOT | CC |
| 25 | Cotton | CCUDCTER | CCUDCTTR | NYBOT | CT |
| 26 | Lean Hogs | CCUDLHER | CCUDLHTR | CME | LH |
| 27 | Live Cattle | CCUDCLER | CCUDLHTR | CME | LC |
| 28 | Feeder Cattle | CCUDFCER | CCUDFCTR | CME | FC |

**Index Sponsor -** Citigroup Global Markets Limited

**Index Launch Date -** 31 August 2012

**Index Start Date -** 31 December 1998

**Excess Return Index Start Level –** 100

**Total Return Index Start Level –** 100

**Index Base Currency -** U.S. Dollars

**Index Business Day -** Each Open Business Day of the New York floor of the CME Group

### Input Files (Master Mapping)

**Input Files for Composition Calculation:**

**For Excess Return Indices:**

**C\_VTDailyPrices\_<VTID\_Date> :**

1. **Required Fields** **:**
   1. Price (Float)
   2. PriceDate (Text convertible to Date)
   3. SettlementDate (Text convertible to Date)
   4. OpenDate (Text convertible to Date)
   5. SpecificInstrumentVTId (Text).
2. A Dataframe (df\_PEFC) is generated using this file and used in Constituent Selection Procedure on RebalDate

**C\_VTIndexSpecificData\_<VTID\_Date> :**

1. **Required Fields** :
   1. RebalanceDate (Text convertible to Date)
   2. PriceDate (Text convertible to Date)
   3. IndexLevel(Float)
   4. IsActive(true / false)
   5. IsActiveLastRebal(true / false)
   6. SpecificInstrumentVTId(Text)
   7. GenericInstrumentVTId(Text)
   8. SpecificTicker(Text)
2. Data from above mention Dataframe (df\_PEFC) for Selection Procedure and the data from this file are used for creating P\_VTIndexSpecificData\_<VTID\_Date> file (with new constituents, if Rebal)

**C\_VTOpenComposition\_<VTID\_Date> :**

1. **Required Fields** **:**
   1. SpecificTicker (Text)
   2. InstrumentVTId (Text)
   3. Units (Float)
   4. PriceDate (Text convertible to Date)
2. This used as template for calculating the NewCloseComposition which is used for P\_VTCloseComposition\_<VTID\_Date> File and NewOpenComposition which is used for P\_VTOpenComposition\_<VTID\_Date> File.
3. For the NewCloseComposition the Constituents Prices and Units are taken from this File while Cash Price and Units are calculated.
4. For NewOpenComposition the Constituent Prices are used while all the Constituent and Cash Uints are calculated according to the Methodology.

**C\_VTCloseComposition\_<VTID\_Date> :**

1. **Required Fields** **[]**
2. For this Index this file is not used.

**C\_VTPotentialEligibleContracts\_<VTID\_Date>:**

1. **Required Fields** **:**
   1. PriceDate (Text convertible to Date)
   2. InstrumentVTId (Text)
2. This File lists the Potential Eligible Contracts for the Month of the RunDate.

**C\_VTPriorRollPrices\_<VTID\_Date> :**

1. **Required Fields** **:**
   1. PriceDate (Text convertible to Date)
   2. SettlementDate(Text convertible to Date)
   3. SpecificInstrumentVTId(Text)
2. This file fetches the prices of all the Constituents required for Prior Roll Yield and Current Roll Yield calculation. It is used in creating a DataFrame (temp\_df\_prior) which is used for calculating Final Settlement Prices.

**C\_VTHolidayCalendar\_<VTID\_Date>:**

1. **Required Fields** **:**
   1. Date (Text convertible to Date)
   2. IsActualHoliday (Boolean(True/False))
2. This File provides the Dates when any of the exchanges has a Holiday. For a particular commodity index there are two types exchanges defined. One being the Fixed where the index is actually listed and declares a holiday when the exchange declares a holiday, the other being the Observant where any of the other indices of the family are listed and when such exchange declares a holiday it is considered as Observant Holiday and used in calculation of All Exchange Open Business Day(used for Rolling)

**For Total Return Indices:**

**C\_VTDailyPrices\_<VTID\_Date>**

1. **Required Fields** **:**
   1. Price (Float)
   2. PriceDate (Text convertible to Date)
   3. SettlementDate (Text convertible to Date)
   4. OpenDate (Text convertible to Date)
   5. SpecificInstrumentVTId (Text).
   6. IndexVTId (Text) – VTId of the corresponding Excess Return Index
2. This file provides the Prices of the constituents and Index Level for both RunDate and LastRunDate of the corresponding Excess Return Index.
3. This data is mainly used for the calculation of Total Return Index Level and also both Close and Open Cash Units.

**C\_VTIndexSpecificData\_<VTID\_Date>**

1. **Required Fields** **:**
   1. PriceDate (Text convertible to Date)
   2. IndexLevel (Float)
   3. FundingRate (Float)
2. This Provides Total Return Level for the LastRunDate and Fuding Rate for both RunDate and LastRunDate.
3. This Funding Rate and IndexLevel Data is used in calculation for IndexLevel(RunDate) and the Constituent Units (both Close and Open)
4. This File is also used in generating OpenIndexSpecificData

**C\_VTOpenComposition\_<VTID\_Date>**

1. **Required Fields** **:**
   1. SpecificTicker (Text)
   2. InstrumentVTId (Text)
   3. Units (Float)
   4. PriceDate (Text convertible to Date)
2. This used as template for calculating the NewCloseComposition which is used for P\_VTCloseComposition\_<VTID\_Date> File and NewOpenComposition which is used for P\_VTOpenComposition\_<VTID\_Date> File.
3. For the NewCloseComposition the Constituents Prices and Units are taken from this File while Cash Price and Units are calculated.
4. For NewOpenComposition the Constituent Prices are used while all the Constituent and Cash Uints are calculated according to the Methodology.

**C\_VTCloseComposition\_<VTID\_Date>**

1. **Required Fields** **[]**
2. For this Index this file is not used.

**C\_VTUnderlyingCloseComposition\_<VTID\_Date>**

1. **Required Fields** **:**
   1. InstrumentVTId (Text)
   2. Units (Float)
2. The Close Units of the constituents from the Excess Return Index(Underlying for Total Return Index) are used to calculate the close units for corresponding constituents in Total Return Index according to the methodolgy

**C\_VTUnderlyingOpenComposition\_<VTID\_Date>**

1. **Required Fields** **:**
   1. InstrumentVTId (Text)
   2. Units (Float)
2. The Close Units of the constituents from the Excess Return Index(Underlying for Total Return Index) are used to calculate the close units for corresponding constituents in Total Return Index according to the methodolgy

**Input Files for Level Calculation:**

**For Excess Return Indices:**

**C\_ CloseCompositionForLevel\_<VTID\_Date>**

1. **Required Fields** **:**
   1. Units (Float)
   2. Price (Float)
   3. FXRate (Float)
   4. PriceFactor (Float)
   5. FxFactor (Float)
2. SumProduct of the above mentioned Fields gives the Index Level which is published in P\_IndexLevel\_<VTID\_Date> File

**For Total Return Indices:**

**C\_ CloseCompositionForLevel\_<VTID\_Date>**

1. **Required Fields** **:**
   1. Units (Float)
   2. Price (Float)
   3. FXRate (Float)
   4. PriceFactor (Float)
   5. FxFactor (Float)
2. SumProduct of the above mentioned Fields gives the Index Level which is published in P\_IndexLevel\_<VTID\_Date> File

### Output Files:

**Output Files for Composition Calculation:**

**For Excess Return Indices:**

**P\_VTIndexSpecificData\_<VTID\_Date>:**

1. **Required Fields** :
   1. RebalanceDate (Text convertible to Date)
   2. PriceDate (Text convertible to Date)
   3. IndexLevel(Float)
   4. IsActive(true / false)
   5. IsActiveLastRebal(true / false)
   6. SpecificInstrumentVTId(Text)
   7. GenericInstrumentVTId(Text)
   8. SpecificTicker(Text)

**P\_VTOpenComposition\_<VTID\_Date>:**

1. **Required Fields** **:**
   1. SpecificTicker (Text)
   2. InstrumentVTId (Text)
   3. Units (Float)
   4. PriceDate (Text convertible to Date)

**P\_VTCloseComposition\_<VTID\_Date>:**

1. **Required Fields** **:**
   1. SpecificTicker (Text)
   2. InstrumentVTId (Text)
   3. Units (Float)
   4. PriceDate (Text convertible to Date)

**For Total Return Indices:**

**P\_VTIndexSpecificData\_<VTID\_Date>:**

1. **Required Fields** **:**
   1. PriceDate (Text convertible to Date)
   2. IndexLevel (Float)
   3. FundingRate (Float)

**P\_VTOpenComposition\_<VTID\_Date>:**

1. **Required Fields** **:**
   1. SpecificTicker (Text)
   2. InstrumentVTId (Text)
   3. Units (Float)
   4. PriceDate (Text convertible to Date)

**P\_VTCloseComposition\_<VTID\_Date>:**

1. **Required Fields** **:**
   1. SpecificTicker (Text)
   2. InstrumentVTId (Text)
   3. Units (Float)
   4. PriceDate (Text convertible to Date)

**Output Files for Level Calculation:**

**For Excess Return Indices:**

**P\_IndexLevel\_<VTID\_Date>:**

1. **Required Fields** **:**
   1. Date (Text convertible to Date)
   2. IndexVTId (Text)
   3. IndexLevel (Float)

**For Total Return Indices:**

**P\_IndexLevel\_<VTID\_Date>:**

1. **Required Fields** **:**
   1. Date (Text convertible to Date)
   2. IndexVTId (Text)
   3. IndexLevel (Float)

### Class – Python

**Excess Return Indices**:

**CUBESDIndexCalculation** : All the Business calculations for the above mentioned **Excess Return** Indices are done here. Further more several Methods and helper Functions are defined for the calculation.

**Methods used for generating the Output Files** :

**Composition Files** :

1. **fn\_VTOpenComp** : Generates
   1. P\_VTIndexSpecificData\_<VTID\_Date>
   2. P\_VTOpenComposition\_<VTID\_Date> Files
2. **fn\_VTCloseComp :** Generates
   1. P\_VTCloseComposition\_<VTID\_Date> File

**Index Level File**

1. **fn\_StdComp2Level :** Generates P\_IndexLevel\_\_<VTID\_Date> using the Input File

C\_ CloseCompositionForLevel\_<VTID\_Date>.

Index Level is the sumproduct of the following fields :

1. Units
2. Price
3. FXRate
4. PriceFactor
5. FxFactor
6. **fn\_VTOpenComp** : This Method calls Four helper Functions / Methods.
   1. **fn\_CUBESDRebalCheck** : Checks if the RunDate is RebalDate
   2. **fn\_CUBESDConstituentSelection :** This Method is used for the Selection of the constituents on Rebalance Date. The Input Data used are
      * 1. C\_VTDailyPrices\_<VTID\_Date>
        2. C\_VTPotentialEligibleContracts\_<VTID\_Date>
        3. C\_VTPriorRollPrices\_<VTID\_Date>.

Helper Fuctions used for Selection:

* + 1. **fn\_FinalPriceCalculator** : This Helper function is used to calculate the Settlement Prices of required commodities for selection procedure. It calls another Two Helper fuctions **fn\_HelperMonthCount** and **fn\_HelperHypotheticalPrice.**
       - 1. **fn\_HelperHypotheticalPrice:** This Helper function is used to calculate Hypothetical Prices for the commodities which are not monthly traded and required for Selection Procedure.
    2. **fn\_HelperMonthCount** : This function is used to calculate the month count of the commodity, nearest being the first Potential Eligible Contract to expiry
  1. **fn\_CUBESDOpenIndexSpecificData** : This method uses data from C\_VTIndexSpecificData\_<VTID\_Date> File and the selected constituents (if there is a Rebal) from **fn\_CUBESDConstituentSelection** and creates the Index specific data for the Open Date
  2. **fn\_CUBESDOpenComposition :** This Method is used to calculate the Open Composition. Inputs taken from **fn\_CUBESDOpenIndexSpecificData** and C\_VTDailyPrices\_<VTID\_Date> to calculate Open Compostion.
     1. **fn\_HelperAEOBDCount :** AEOBD – All Exchange Open Business Day. After Rebal the rolling takes place till All the Exchanges are open for atleast 3 days. This Helper function calculates the count of the number of days all exchanges are open after Rebal Date.

Based on AEOBD count the formula for calculation of open units of the constituents changes as per the Rule Book. Cash Units are calculated using the Index Level calculated in **fn\_CUBESDCalculateIndexLevel.**

1. **fn\_VTCloseComp** : This Method calls Three helper Functions / Methods.
   1. **fn\_CUBESDCloseComposition :** This generates New close composition for the day using C\_VTOpenComposition\_<VTID\_Date> File which contains Open compotion for the Last RunDate.
   2. **fn\_CUBESDCalculateIndexLevel :** 
      1. This Method takes data from C\_VTIndexSpecificData\_<VTID\_Date> (Active Constituents) and C\_VTDailyPrices\_<VTID\_Date> (Settlement prices of the Active and LastReabalActive Constituents).
      2. The Helper function **fn\_HelperAEOBDCount** is used to calculate the AEOBD count based on which the formula for IndexLevel calculation changes.
   3. **fn\_CUBESDCalculateCloseCashUnits :** Takes data from **fn\_CUBESDCloseComposition** i.e., New Close Composition generated from the C\_VTOpenComposition\_<VTID\_Date> File and calculates the Close Cash Units.

**Total Return Indices:**

**CUBESDTRIndexCalculation :** All the Business calculations for the above mentioned(in the Table) **Total Return** Indices are done here. Further more several Methods and helper Functions are defined for the calculation.

**Methods used for generating the Output Files** :

**Composition Files** :

1. **fn\_VTOpenComp** : Generates
   1. P\_VTIndexSpecificData\_<VTID\_Date>
   2. P\_VTOpenComposition\_<VTID\_Date> Files
2. **fn\_VTCloseComp :** Generates
   1. P\_VTCloseComposition\_<VTID\_Date> File

**Index Level File**

1. **fn\_StdComp2Level :** Generates P\_IndexLevel\_\_<VTID\_Date> using the Input File

C\_ CloseCompositionForLevel\_<VTID\_Date>.

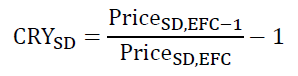
Index Level is the sumproduct of the following fields :

1. Units
2. Price
3. FXRate
4. PriceFactor
5. FxFactor
6. **fn\_VTOpenComp** : This Method calls Two helper Functions / Methods.
   1. **fn\_CUBESDTROpenIndexSpecificData :** This Method takes data from the Input File C\_VTIndexSpecificData\_<VTID\_Date> and creates OpenIndexSpecificData where the calculated IndexLevel for RunDate is filled.Output : P\_VTIndexSpecificData\_<VTID\_Date>
   2. **fn\_CUBESDTROpenComposition :**
      1. This Function takes data from
         1. C\_VTOpenComposition\_<VTID\_Date>
         2. C\_VTUnderlyingOpenComposition\_<VTID\_Date>
         3. C\_VTDailyPrices\_<VTID\_Date>
         4. C\_VTIndexSpecificData\_<VTID\_Date>
      2. Here the calculation of Constituent and Cash Units are calculated according to the methodology for Total Return Indices and the OpenComposition for the RunDate is published through P\_VTOpenComposition\_<VTID\_Date> File
7. **fn\_VTCloseComp :** This Method calls Two helper Functions / Methods.
   1. **fn\_CUBESDTRCloseComposition :** 
      1. This Function takes data from
         1. C\_VTOpenComposition\_<VTID\_Date>
         2. C\_VTUnderlyingCloseComposition\_<VTID\_Date>
         3. C\_VTDailyPrices\_<VTID\_Date>
         4. C\_VTIndexSpecificData\_<VTID\_Date>
      2. Here the calculation of Constituent Units are calculated(same as the units flowing in C\_VTOpenComposition\_<VTID\_Date> File) according to the methodology for Total Return Indices and the CloseComposition for the RunDate is published through P\_VTCloseComposition\_<VTID\_Date> File
   2. **fn\_CUBESDTRCalculateIndexLevel :** 
      1. This Function takes data from
         1. C\_VTOpenComposition\_<VTID\_Date>
         2. C\_VTDailyPrices\_<VTID\_Date>
         3. C\_VTIndexSpecificData\_<VTID\_Date>
      2. Here the Index Level for the RunDate and Close Cash Units are calculated. OutPut File : P\_VTCloseComposition\_<VTID\_Date>

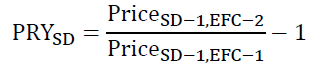
### Business Logic – Index Calculation

**Excess Return Indices:** Majorly contains two procedures for Index Calculation.

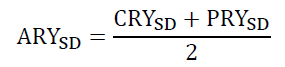
1. **Selection Procedure** :
   1. **Target**: Select three contracts from the Eligible Contracts (Potential Eligible Contracts which are listed and traded on the exchange)
   2. **Procedure**:
      1. **Step1**: Calculation of CRY (Current Roll Yield)



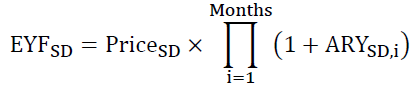
* + 1. **Step2:** Calculation of PRY (Prior Roll Yield)



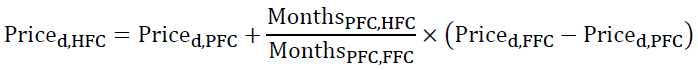
* + 1. **Step3:** Calculation of ARY (Average Roll Yield)



* + 1. **Step4:** Calculation of EYF (Expected Yield Factor)



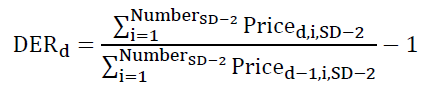
* + 1. The Selected Futures Contracts determined in respect of each Selection Day shall be the three Eligible Futures Contracts in respect of SD the Expected Yield Factors of which are the lowest
  1. **Settlement Prices:** Settlement Prices as published by the Exchange on a business day.
     1. **Excluded Futures Contracts**: Hypothetical Futures Contract determined in respect of such Excluded Futures Contract
     2. **Price of a Hypothetical Futures Contract:**



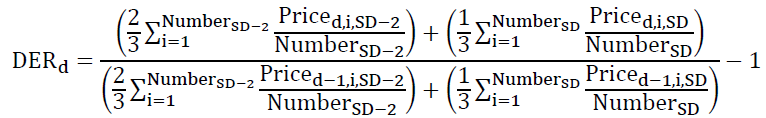
1. **Composition and Index Calculation** :
   1. **Index Level**:



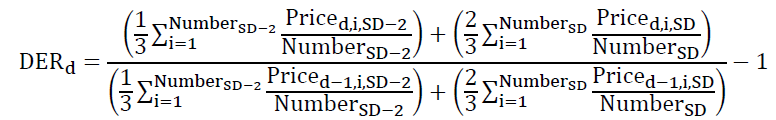
* 1. **Daily Excess Return(DER):** Rolling takes place for first three all exchange open business days (AEOBD). Based on the count of AEOBD there are four formulae for calculation of DER.
     1. **AEOBD <= 1:**



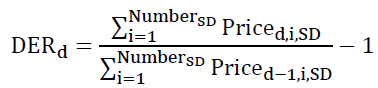
* + 1. **AEOBD = 2:**



* + 1. **AEOBD = 3:**



* + 1. **AEOBD > 3:**



1. **Units or Composition :**
   * 1. **Constituent Open Units** :

(Index Level(t)/Sum of Settlement Prices(t) of the (three) selected constituents)

* + 1. **Cash Open Units** :

Index Level(t) – Sumproduct(Constituent Open Units(t),Settlement Prices(t))

* + 1. **Constituent Close Units** :

These are the same units we receive from the C\_VTOpenComposition\_<VTID\_Date> File.

* + 1. **Cash Close Units** :

Index Level(t) – Sumproduct(Constituent Close Units(t),Settlement Prices(t))

**Total Return Indices**

1. **Index Level Calculation :**

As the constituents for the Total Return Index will be same as Excess Return Index there is no different Selection procedure for Total Return Indices.

The Calculation of the Total Return Indices requires following data.

IL(t) : Index Level on RunDate

IL(t-1) : Index Level on Last RunDate

ERILd : Excess Return Index Level on RunDate

ERILd-1 : Excess Return Index Level on LastRunDate

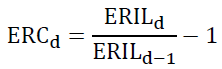
TRILd – Total Return Index Level on RunDate

TRILd-1 – Total Return Index Level on LastRunDate

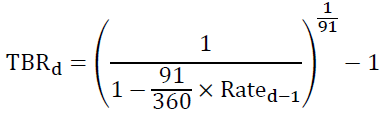
Rated-1 : Funding Rate (USB3MTA) on LastRunDate

Rated : Funding Rate (USB3MTA) on RunDate

1. **Index Level :**
   1. IL(t) = IL(t-1)\*(1+ ERC(t))\*(1+TBR(t))^(NextOpenDate – RunDate)
2. **ERC :**



1. **TBR :**



1. **Units or Composition :**
   * 1. **Constituent Open Units** :

UnderlyingOpenUnits\*( (TRILd-1/ ERILd-1 )\*(1/(1-(91/360)\*( Rated/100)))^((OpenDate – RunDate)/91)

* + 1. **Cash Open Units** :

Index Level(t) – Sumproduct(Constituent Open Units(t),Settlement Prices(t))

* + 1. **Constituent Close Units** :

UnderlyingCloseUnits\*( (TRILd-1/ ERILd-1 )\*(1/(1-(91/360)\*( Rated-1 /100)))^((RunDate – LastRunDate)/91)

* + 1. **Cash Close Units** :

Index Level(t) – Sumproduct(Constituent Close Units(t),Settlement Prices(t))

## Index – Introduction

**Name** : Europe Style Indices

**Type** : Equities

**List of Indices** :

Ei:EU:XX:XX:X:XX:SGET:XXXX:ML

Ei:EU:XX:XX:X:XX:SMET:XXXX:ML

Ei:EU:XX:XX:X:XX:SSET:XXXX:ML

Ei:EU:XX:XX:X:XX:SVET:XXXX:ML

Ei:EU:XX:XX:X:XX:SWET:XXXX:ML

**Index Sponsor** : Bank of America , Merrill Lynch

**Index Base Currency** : EUR Dollar

### Input Files

### Input Files for Composition Calculation:

* 1. C\_BankComposition\_<VTId>\_<RunDate>
     1. Required Fields
        1. OpenUnitsAdjusted : Float
        2. InstrumentVTId : Text
        3. PriceInInstrumentCurrency : Float
        4. FxRate : Float
  2. C\_OpenCompositions\_<VTId>\_<RunDate>
     1. Required Fields
        1. InstrumentVTId : Text
        2. Units : Float
        3. PriceDate : Text convertible to Date
        4. IndexVTId : Text
        5. IsDelisted : Boolean (True/False)
        6. PriceInInstrumentCurrency : Float
        7. FxRate : Float
  3. C\_CorporateActions\_<VTId>\_<RunDate>
     1. Required Fields
        1. InstrumentVTId : Text
        2. ShareAdjustmentFactor : Float
        3. PriceAdjustmentFactor : Float
        4. DividendType : Text
        5. CashAmount : Float
        6. Factor : Float
        7. IssuePrice : Float
  4. C\_IndexCalculationDetails\_<VTId>\_<RunDate>
     1. Required Fields
        1. IsRebalance : Boolean (True/False)
        2. OpenDivisor : Float
        3. OpenTransactionCost : Float
        4. PriceDate : Text convertible to Date
        5. Divisor : Float
        6. IndexLevel : Float
        7. TransactionCost : Float

**Input Files for Level Calculation:**

* 1. C\_CloseCompositionForLevel\_<VTId>\_<RunDate>
     1. Required Fields
        1. Units : Float
        2. PriceInInstrumentCurrency : Float
        3. FXRate : Float
  2. C\_IndexCalculationDetailsForLevel\_<VTId>\_<RunDate>
     1. Required Fields
        1. IsRebalance : Boolean (True/False)
        2. OpenDivisor : Float
        3. OpenTransactionCost : Float
        4. PriceDate : Text convertible to Date
        5. Divisor : Float
        6. IndexLevel : Float
        7. TransactionCost : Float

### Output Files

#### Output Files for Composition Calculation:

* 1. P\_VTCloseComposition\_<VTId>\_<RunDate>
     1. Required Fields
        1. InstrumentVTId : Text
        2. Units : Float
        3. PriceDate : Text convertible to Date
        4. IndexVTId : Text
        5. PriceInInstrumentCurrency : Float
        6. FxRate : Float
  2. P\_VTOpenComposition\_<VTId>\_<RunDate>
     1. Required Fields
        1. InstrumentVTId : Text
        2. Units : Float
        3. PriceDate : Text convertible to Date
        4. IndexVTId : Text
        5. IsDelisted : Boolean (True/False)
        6. PriceInInstrumentCurrency : Float
        7. FxRate : Float
  3. P\_VTIndexCalculationDetails\_<VTId>\_<RunDate>
     1. Required Fields
        1. IsRebalance : Boolean (True/False)
        2. OpenDivisor : Float
        3. OpenTransactionCost : Float
        4. PriceDate : Text convertible to Date
        5. Divisor : Float
        6. IndexLevel : Float
        7. TransactionCost : Float

**Output Files for Level Calculation:**

* 1. P\_IndexLevel\_<VTId>\_<RunDate>
     1. Required Fields
        1. Date (Text convertible to Date)
        2. IndexVTId (Text)
        3. IndexLevel (Float)

### Class – Python

**EquityIndexCalculation** : This is the VT class under which the calculations for Europe Style Indices are coded

* 1. Required parameters for the Initialization of the class
     1. **RunDate** : Date for which the Index level is calculated (from C# argument)
     2. **LastRunDate** : Date previous to the Run Date (from C# argument)
     3. **OpenCompositions** : Data collected from the input file C\_OpenCompositions\_<VTId>\_<RunDate>
     4. **BankComposition** : Data collected from the input file C\_BankComposition\_<VTId>\_<RunDate>
     5. **CorporateActions** : Data collected from the input file C\_CorporateActions\_<VTId>\_<RunDate>
     6. **IndexCalculationDetails** : Data collected from the input file C\_IndexCalculationDetails\_<VTId>\_<RunDate>
     7. **NewOpenComposition** : Dataframe inititated similar to OpenCompositions
     8. **NewCloseComposition** : Dataframe inititated similar to OpenCompositions
     9. **NewIndexCalculationDetails** : A copy of IndexCalculationDetails
     10. **IndexLevel** : (=0) Initiated from zero
     11. **temp\_df\_CA** : Empty Dataframe
     12. **directory** : Directory where the input and output files are stored
     13. **IndexVTId** : VTId of the Index (from C# argument)
     14. **log** : plogger is used for logging the processes
  2. According to the Master Mapping there is one sub-process called for this **VtCompOpenClose** which calls two functions from the VT class, **fn\_VTCloseComp** and **fn\_VTOpenComp**
     1. **fn\_VTCloseComp** : The output file **P\_VTCloseComposition\_<VTId>\_<RunDate>** is generated here. Also the close composition cash is calculated here. At first from the OpenCompositions delisted instruments and the existing cash instrument are separated and copied to NewCloseComposition dataframe. The delisted instruments data is used to calculate the close cash and atlast appended to the NewCloseComposition. Now the following method is called
        1. **fn\_rebalCheckCloseComp()** :
           1. **If Rebalance :** If there is a rebalance then all the instruments from BankComposition whose OpenUnitsAdjusted are non-zero are copied and those that are not part of NewCloseComposition are filtered out. These are appended to NewCloseComposition for which units are taken as zero and the prices and FX rates are as per the BankComposition
           2. **If NO Rebalance** : The NewCloseComposition remains same.
     2. **fn\_VTOpenComp** : The output files **P\_VTOpenComposition\_<VTId>\_<RunDate>** and **P\_VTIndexCalculationDetails\_<VTId>\_<RunDate>** are generated here. This function calls for four more methods
        1. **fn\_CAHandling()** : The Corporate Actions we get in the input file **C\_CorporateActions\_<VTId>\_<RunDate>** are handled here and the Price and Units are adjusted accordingly in the OpenComposition. Targer is to calculate PAF (Price Adjustement Factor) which is multiplied to the open price to adjuct CA. Similarly SAF (Share Adjustment Factor) is calculated and multiplied to the Open Units. It is necessary that SAF and PAF to be null in the file otherwise it is considered they are entered manually and the values are directly used. SAF and PAF are calculated are stored SAF and PAF formulae for different CA’s (Dividend Type) are mentioned below.
           1. Cash Dividend or Special Cash Dividend :

SAF = 1

PAF =  [Closing Price  - ((Cash Amount(Cash Dividend) + Cash Amount(Special Dividend))\*(1-withholding tax))]/Closing price

* + - * 1. Stock Split :

SAF = Factor

PAF = 1/Factor

* + - * 1. Stock Dividend :

SAF = 1 + Factor

PAF = 1/(1+Factor)

* + - * 1. Rights Offering :

SAF = (1+Factor)

PAF = (Close Price + Issue Price\*FX(to covert form Issue currency to Underlying Currency)\*Factor)/((1+Factor)\*Close Price)

* + - 1. **fn\_rebalCheckOpenComp()** : The Open Composition is calculated here. NewOpenComposition is the dataframe used for Open Composition calculations which is a copy of the NewCloseComposition.The SAF and PAF are assigned as 1 for initialization. This handles both the cases Rebalance and without Rebalance
         1. **If Rebalance** : The Units in the NewOpenComposition is copied as the OpenUnitsAdjusted from the BankComposition. And the prices are adjusted with PAF. Cash Units are defaulted to zero.
         2. **If NO Rebalance** : The Units and Prices in NewOpenComposition are adjusted with SAF and PAF respectively.
      2. **fn\_CalculateIndexLevel() :** Index level for the Run Date is calculated here. For the Index calculation purposes we need close divisor and close transaction cost which are taken from the C\_IndexCalculationDetails\_<VTId>\_<RunDate> file. Formula for Index Calculation. The following data taken from NewCloseComposition

**(Units\*** **PriceInInstrumentCurrency\*** **FxRate)/** **CloseDivisor-TransactionCost**

* + - 1. **fn\_IndexCalculationDetails()** : The Index calculation details for the open date are calculated here. This is output for P\_VTIndexCalculationDetails\_<VTId>\_<RunDate>.

CloseSumproduct = sum(Units\* PriceInInstrumentCurrency\* FxRate) (NewCloseComposition)

OpenSumproduct = sum(Units\* PriceInInstrumentCurrency\* FxRate) (NewOpenComposition)

* + - * 1. **if NO Rebalance :**

**OpenDivisor =** (CloseDivisor \* (IndexLevel + CloseTransactionCost) + (OpenSumproduct - CloseSumproduct)) / (IndexLevel + CloseTransactionCost)

**OpenTransactionCost =** CloseTransactionCost

* + - * 1. **If Rebalance :**

**OpenDivisor** = (CloseDivisor \* (IndexLevel + CloseTransactionCost) + (OpenSumproduct - CloseSumproduct)) / IndexLevel

**OpenTransactionCost** = sum(abs(NewCloseComposition["Units"]- NewOpenComposition["Units"])\*NewCloseComposition["PriceInInstrumentCurrency"]\*NewCloseComposition["FxRate"])\*Fee

1. **fn\_EqComp2Level:** Generates P\_IndexLevel\_\_<VTID\_Date> using the Input File

C\_ CloseCompositionForLevel\_<VTID\_Date>.

Index Level is the sumproduct of the following fields : SumProduct

* 1. Units
  2. PriceInInstrumentCurrency
  3. FXRate

IndexLevel = (SumProduct/Divisor) - TransactionCost

## Index – Introduction

**Name** : BofAML Investable European 4-Year Constant Tenor Short Dividend Futures Index

**Type** : Futures

**Index VTId** : Ei:EU:DT:XX:X:XX:SDIV:XXXX:ML

**Index Sponsor** : Bank of America , Merrill Lynch

**Index Base Date** : 30 June 2008

**Index Base Currency** : EUR Dollar

### Input Files

### Input Files for Composition Calculation:

1. C\_VTIndexSpecificData\_<VTId>\_<RunDate>
2. Required Fields
   1. IndexLevelRebal - Float
   2. GenericTicker - Text
   3. Weights - Float
   4. SpecificTicker - Text
   5. PriceDate – Text convertible to date
   6. Price - Float
   7. RebalancingTransactionCost - Float
   8. SpecificInstrumentVTId - Text
   9. SwitchRollFlag – Boolean (0 or 1)
   10. Units - Float
3. C\_VTOpenComposition\_<VTId>\_<RunDate>
4. Required Fields
   1. GenericTicker - Text
   2. SpecificTicker - Text
   3. InstrumentVTId – Text
   4. Units - Float
5. C\_VTDailyPrices\_<VTId>\_<RunDate>
6. Required Fields
   1. PriceDate – Text convertible to date
   2. OpenDate - Text convertible to date
   3. GenericTicker - Text
   4. SettlementDate - Text convertible to date
   5. SpecificTicker - Text
   6. Price - Float
   7. SpecificInstrumentVTId - Text

**Input Files for Level Calculation:**

1. C\_CloseCompositionForLevel\_<VTId>\_<RunDate>
   * 1. Required Fields
        1. Units (Float)
        2. Price (Float)
        3. FXRate (Float)
        4. PriceFactor (Float)
        5. FxFactor (Float)
     2. SumProduct of the above mentioned Fields gives the Index Level which is published in P\_IndexLevel\_<VTID\_Date> File

### Output Files

#### Output Files for Composition Calculation:

1. P\_VTCloseComposition\_<VTId>\_<RunDate>
2. Required Fields
   * + 1. GenericTicker - Text
       2. SpecificTicker - Text
       3. InstrumentVTId – Text
       4. Units - Float
3. P\_VTOpenComposition\_<VTId>\_<RunDate>
4. Required Fields
   * + 1. GenericTicker - Text
       2. SpecificTicker - Text
       3. InstrumentVTId – Text
       4. Units - Float
5. P\_VTIndexSpecificData\_<VTId>\_<RunDate>
6. Required Fields
   * + 1. IndexLevelRebal - Float
       2. GenericTicker - Text
       3. Weights - Float
       4. SpecificTicker - Text
       5. PriceDate – Text convertible to date
       6. Price - Float
       7. RebalancingTransactionCost - Float
       8. SpecificInstrumentVTId - Text
       9. SwitchRollFlag – Boolean (0 or 1)
       10. Units - Float

**Output Files for Level Calculation:**

* 1. P\_IndexLevel\_<VTId>\_<RunDate>
     1. Required Fields
        1. Date (Text convertible to Date)
        2. IndexVTId (Text)
        3. IndexLevel (Float)

### Class – Python

**SDIVIndexCalculation** : This is the VT class under which the calculations for

MLEISDIV are coded

* 1. Required parameters for the Initialization of the class
     1. **RunDate** : Date for which the Index level is calculated (from C# argument)
     2. **LastRunDate** : Date previous to the Run Date (from C# argument)
     3. **OpenDate** : The next business Open day for trading
     4. **DailyPrices** : Data collected from the input file C\_VTDailyPrices\_<VTId>\_<RunDate>
     5. **IndexSpecificData** : Data collected from the input file C\_VTOpenComposition\_<VTId>\_<RunDate>
     6. **OpenComposition** : Data collected from the input file C\_VTOpenComposition\_<VTId>\_<RunDate>
     7. **RebalFlag :** (=0) Thursday or if such Thursday is not an Index Valuation Date, the Index Valuation Date immediately preceding such Thursday
     8. **SwitchRollFlag :** (=0) usually the Thursday prior to the third Friday in December
     9. **NewCloseComposition** : Dataframe inititated similar to OpenComposition
     10. **OpenIndexSpecificData** : Dataframe inititated similar to IndexSpecificData
     11. **NewOpenComposition** : Dataframe inititated similar to OpenComposition
     12. **IndexLevel** : (=0) Initiated from zero
     13. **directory** : Directory where the input and output files are stored
     14. **IndexVTId** : VTId of the Index (from C# argument)
     15. **log** : plogger is used for logging the processes
  2. According to the Master Mapping there is one sub-process called for this **VtCompOpenClose** which calls two functions from the VT class, **fn\_VTCloseComp** and **fn\_VTOpenComp**
     1. **fn\_VTCloseComp** : The output file **P\_VTCloseComposition\_<VTId>\_<RunDate>** is generated here. The following methods are called
        1. **fn\_SDIVRebalCheck** : Here the RunDate is checked for Roll and Switch Roll Flag. If the Open Date is the settlement date for the nearest to maturity contract the such Roll date is Switch Roll Date
        2. **fn\_SDIVCloseComposition** : Here the **OpenComposition** dataframe is copied to **NewCloseComposition** without the cash units for close composition and date is changed to RunDate
        3. **fn\_SDIVCalculateIndexLevel** : This method takes input data from the files C\_VTDailyPrices\_<VTId>\_<RunDate> and C\_VTIndexSpecificData\_<VTId>\_<RunDate>. Here the calculation of Index level is done on the basis of Business logic mentioned in Rule book. Explained in Business Logic section
        4. **fn\_SDIVCalculateCashUnits** : This method takes input data from the files C\_VTDailyPrices\_<VTId>\_<RunDate> and C\_VTIndexSpecificData\_<VTId>\_<RunDate> and also uses the **NewCloseComposition.** Here the calculation of Close Cash Units is done. Explained in Business Logic section
     2. **fn\_VTOpenComp** : The output files **P\_VTOpenComposition\_<VTId>\_<RunDate>** and **P\_VTIndexCalculationDetails\_<VTId>\_<RunDate>** are generated here. This function calls for two more methods
        1. **fn\_SDIVOpenIndexSpecificData** : This method takes input data from the files C\_VTDailyPrices\_<VTId>\_<RunDate> and C\_VTIndexSpecificData\_<VTId>\_<RunDate> Here the Index Specific Data is modified as per the instructions of Roll and Switch Roll. The Weights and Units are calculated whenever there is a Roll or Switch Roll. Explained in the Business Logic
        2. **fn\_SDIVOpenComposition** : This method takes input data from the files C\_VTDailyPrices\_<VTId>\_<RunDate> and the OpenIndexSpecificData to prepare open composition. NewOpenComposition is created by copying the OpenComposition and changing the constituents if there are any due to Roll or Switch Roll. The Open Cash units are also calculated here

**Index Level File**

1. **fn\_StdComp2Level :** Generates P\_IndexLevel\_\_<VTID\_Date> using the Input File

C\_ CloseCompositionForLevel\_<VTID\_Date>.

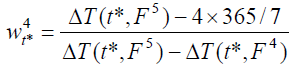
Index Level is the sumproduct of the following fields :

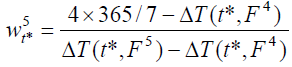
1. Units
2. Price
3. FXRate
4. PriceFactor
5. FxFactor

### Business Logic – Index Calculation :

This Index replicates a short position in a synthetic 4-year constant tenor Contract by rolling between Contracts 1. Fourth closest to Maturity and 2. Fifth closest to Maturity. The Index is designed to provide an excess return

1. **Weights Calculation** : Weights are calculated on every Roll Date. Apparently on every Thursday of the week (if Holiday then previous day). Formulae for the weights of the two contracts are





 - The Weight of the Fourth Closest-To-Maturity Contract in respect of Roll Date “t\*”

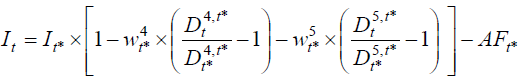
 - The Weight of the Fifth Closest-To-Maturity Contract in respect of Roll Date “t\*”

F4 - The Final Settlement Date of the Fourth Closest-To-Maturity Contract as determined in respect of Roll Date “t\*”

F5 - The Final Settlement Date of the Fifth Closest-To-Maturity Contract as determined in respect of Roll Date “t\*”

∆T(tx , ty) - The number of calendar days from, but excluding, date ”tX” to, but excluding, date” tY”, divided by 7

1. **Index Value Calculation :**



It - The Index Value in respect of Index Valuation Date “t”

I0 - The Index Value in respect of the Index Base Date

t\* - The Roll Date immediately preceding Index Valuation Date “t” (for the avoidance of doubt, if Index Valuation Date “t” is a Roll Date, then t\* refers to the previous Roll Date)

It\* - The Index Value in respect of Roll Date “t\*”

 - The Daily Settlement Price in respect of Index Valuation Date “t” of the Fourth Closest-To-Maturity Contract as determined in respect of Roll Date “t\*”

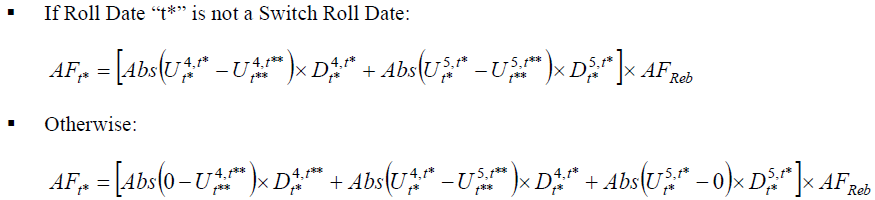
 - The Daily Settlement Price in respect of Roll Date “t\*” of the Fourth Closest-To-Maturity Contract as determined in respect of Roll Date “t\*”

 - The Daily Settlement Price in respect of Index Valuation Date “t” of the Fifth Closest-To-Maturity Contract as determined in respect of Roll Date “t\*”

 - The Daily Settlement Price in respect of Roll Date “t\*” of the Fifth Closest-To-Maturity Contract as determined in respect of Roll Date “t\*” \*tAF

AFt\* - The Adjustment Factor in respect of Roll Date “t\*”, as defined below

1. **Adjustment Factor Calculation** :



AF0 - The Adjustment Factor in respect of the Index Base Date, which equals 0.0

t\*\* - The Roll Date immediately preceding Roll Date “t\*”

 - The number of units in respect of Roll Date “t\*” of the Fourth Closest-To-Maturity Contract as determined in respect of Roll Date “t\*”, as defined below

 - The number of units in respect of Roll Date “t\*\*” of the Fourth Closest-To-Maturity Contract as determined in respect of Roll Date “t\*\*”, as defined below

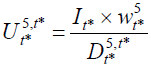
 - The number of units in respect of Roll Date “t\*” of the Fifth Closest-To-Maturity Contract as determined in respect of Roll Date “t\*”, as defined below

 - The number of units in respect of Roll Date “t\*\*” of the Fifth Closest-To-Maturity Contract as determined in respect of Roll Date “t\*\*”, as defined below

AFReb - The Rebalancing Adjustment Factor, which equals 0.50%

1. **Units Calculation** :





1. **Cash Units :**
2. **Cash Open Units** :

Index Level(t) – Sumproduct(Constituent Open Units(t),Settlement Prices(t))

1. **Cash Close Units** :

Index Level(t) – Sumproduct(Constituent Close Units(t),Settlement Prices(t))

## Index – Introduction

**Name** : CS Relative Value Volatility Index Excess Return (“RVOL”)

**Type** : Futures

**Index VTId** : Vi:US:XX:XX:X:XX:RVOL:XXXX:CS

**Index Sponsor** : Credit Suisse International

**Index Base Date** : 16 April 2008

**Index Base Currency** : US Dollar

### Input Files

### Input Files for Composition Calculation:

1. C\_VTIndexSpecificData\_<VTId>\_<RunDate>
2. Required Fields
   1. IndexLevel – Float
   2. PriceDate – Text convertible to date
   3. Signal – Float
   4. LastSignal - Float
   5. Target - Float
   6. GenericTicker - Text
   7. Weights – Float
   8. LastWeights - Float
   9. Units - Float
   10. LastUnits - Float
   11. Cost - Float
   12. Price - Float
   13. Fee - Float
   14. RebalancingTransactionCost – Float
   15. LastRebalancingTransactionCost
3. C\_VTOpenComposition\_<VTId>\_<RunDate>
4. Required Fields
   1. GenericTicker - Text
   2. SpecificTicker – Text
   3. Price - Float
   4. Units - Float
5. C\_VTDailyPrices\_<VTId>\_<RunDate>
6. Required Fields
   1. PriceDate – Text convertible to date
   2. GenericTicker - Text
   3. SpecificTicker - Text
   4. Price - Float

**Input Files for Level Calculation:**

1. C\_CloseCompositionForLevel\_<VTId>\_<RunDate>
   * 1. Required Fields
        1. Units (Float)
        2. Price (Float)
        3. FXRate (Float)
        4. PriceFactor (Float)
        5. FxFactor (Float)
     2. SumProduct of the above mentioned Fields gives the Index Level which is published in P\_IndexLevel\_<VTID\_Date> File

### Output Files

#### Output Files for Composition Calculation:

1. P\_VTCloseComposition\_<VTId>\_<RunDate>
2. Required Fields
   * + 1. GenericTicker - Text
       2. SpecificTicker – Text
       3. Price - Float
       4. Units - Float
3. P\_VTOpenComposition\_<VTId>\_<RunDate>
4. Required Fields
   * + 1. GenericTicker - Text
       2. SpecificTicker – Text
       3. Price - Float
       4. Units - Float
5. P\_VTIndexSpecificData\_<VTId>\_<RunDate>
6. Required Fields
   * + 1. IndexLevel – Float
       2. PriceDate – Text convertible to date
       3. Signal – Float
       4. LastSignal - Float
       5. Target - Float
       6. GenericTicker - Text
       7. Weights – Float
       8. LastWeights - Float
       9. Units - Float
       10. LastUnits - Float
       11. Cost - Float
       12. Price - Float
       13. Fee - Float
       14. RebalancingTransactionCost – Float
       15. LastRebalancingTransactionCost

**Output Files for Level Calculation:**

1. P\_IndexLevel\_<VTId>\_<RunDate>
   * 1. Required Fields
        1. Date (Text convertible to Date)
        2. IndexVTId (Text)
        3. IndexLevel (Float)

### Class – Python

**RVOLIndexCalculation** : This is the VT class under which the calculations for

CSEARVOL are coded

1. Required parameters for the Initialization of the class
2. **RunDate** : Date for which the Index level is calculated (from C# argument)
3. **LastRunDate** : Date previous to the Run Date (from C# argument)
4. **DailyPrices** : Data collected from the input file C\_VTDailyPrices\_<VTId>\_<RunDate>
5. **IndexSpecificData** : Data collected from the input file C\_VTOpenComposition\_<VTId>\_<RunDate>
6. **OpenComposition** : Data collected from the input file C\_VTOpenComposition\_<VTId>\_<RunDate>
7. **NewCloseComposition** : Dataframe inititated similar to OpenComposition
8. **OpenIndexSpecificData** : Dataframe inititated similar to IndexSpecificData
9. **NewOpenComposition** : Dataframe inititated similar to OpenComposition
10. **IndexLevel** : (=0) Initiated from zero
11. **temp\_df :** Initiated a temporary Dataframe
12. **directory** : Directory where the input and output files are stored
13. **IndexVTId** : VTId of the Index (from C# argument)
14. **log** : plogger is used for logging the processes
15. According to the Master Mapping there is one sub-process called for this **VtCompOpenClose** which calls two functions from the VT class, **fn\_VTCloseComp** and **fn\_VTOpenComp**
16. **fn\_VTCloseComp** : The output file **P\_VTCloseComposition\_<VTId>\_<RunDate>** is generated here. The following methods are called
17. **fn\_RVOLCloseComposition**: Here the **OpenComposition** dataframe is copied to **NewCloseComposition** without the cash units for close composition and date is changed to RunDate
18. **fn\_RVOLCalculateIndexLevel** : This method takes input data from the files C\_VTDailyPrices\_<VTId>\_<RunDate>, C\_VTIndexSpecificData\_<VTId>\_<RunDate> and **OpenComposition.** Here the calculation of Index level is done on the basis of Business logic mentioned in Rule book. Explained in Business Logic section
19. **fn\_RVOLCalculateCloseCashUnits** : This method takes input data from the files C\_VTDailyPrices\_<VTId>\_<RunDate> and C\_VTIndexSpecificData\_<VTId>\_<RunDate> and also uses the **NewCloseComposition.** Here the calculation of Close Cash Units is done. Explained in Business Logic section
20. **fn\_VTOpenComp** : The output files **P\_VTOpenComposition\_<VTId>\_<RunDate>** and **P\_VTIndexCalculationDetails\_<VTId>\_<RunDate>** are generated here. This function calls for two more methods
21. **fn\_RVOLOpenIndexSpecificData** : This method takes input data from the files C\_VTDailyPrices\_<VTId>\_<RunDate> and C\_VTIndexSpecificData\_<VTId>\_<RunDate>. Here the Index Specific Data is modified as per the instructions of Rulebook. The Weights, Units, Signal, Cost are calculated for the Open Date. Explained in the Business Logic Section
22. **fn\_RVOLOpenComposition** : Here the **OpenComposition** dataframe is copied to **NewOpenComposition** for open composition and date is changed to RunDate. Open cash units are also calculated and replaced here

**Index Level File**

1. **fn\_StdComp2Level :** Generates P\_IndexLevel\_\_<VTID\_Date> using the Input File

C\_ CloseCompositionForLevel\_<VTID\_Date>.

Index Level is the sumproduct of the following fields :

1. Units
2. Price
3. FXRate
4. PriceFactor
5. FxFactor

### Business Logic – Index Calculation :

This Index is rebalanced everyday

1. **Daily Calculation of the Index** :



“t-1” is the Index Calculation Day preceding the Index Calculation Day t

“Indext-1 ” is the Index Value as of the close of the previous Index Calculation Day, t-1

“nST\_Indext-1” and “nMT\_Indext-1” refers to the number of units held in ST\_Index and MT\_Index, respectively, as of the end of the previous Index Calculation Day, t-1

“ST\_Indext” and “MT\_Indext” , respectively, denote the values of ST\_Index and the MT\_Index as of the close of Index Calculation Day t

“TAt-1“denotes a liquidity Trading Adjustments as defined below

“DAt-1” denotes the Daily Accrual as defined below

1. **Daily Signal :**

Signalt = ST\_Slopet – MT\_Slopet

ST\_Slopet = UX2t – UX1t

MT\_Slopet = UX7t – UX4t

“UX1t” is the closing level of the 1st serial monthly VIX Future on Index Calculation Day, t

“UX2t” is the closing level of the 2nd serial monthly VIX Future on Index Calculation Day, t

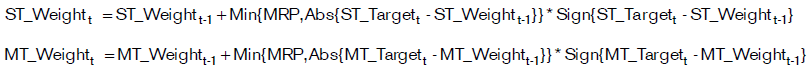
“UX4t” is the closing level of the 4th serial monthly VIX Future on Index Calculation Day, t

“UX7t” is the closing level of the 7th serial monthly VIX Future on Index Calculation Day, t

1. **Target Weights :** Target Weights as of the close of any Index Calculation Day t, which depend on Signalt-1



1. **Actual Weights :**



“t-1” is the Index Calculation Day preceding the Index Calculation Day t

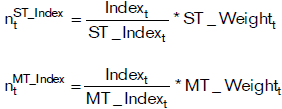
“MRP” is the Maximum Rebalance Percentage, which will be equal to .025

“ST\_Targett” is the Target Weight applicable to the ST\_Index on Index Calculation Day t

“MT\_Targett” is the Target Weight applicable to the MT\_Index on Index Calculation Day t

“Sign(x)” is the *sign* mathematical function, which returns the sign of any real number and is equal to +1 if x is positive and -1 if x is negative.

1. **Index Units :**



“Indext” is the Index Value as of the close of the Index Calculation Day, t

“ST\_Weightt” is the actual weight applicable to ST\_Index for Index Calculation Day, t

“MT\_Weightt” is the actual weight applicable to MT\_Index for Index Calculation Day, t

1. **Trading Adjustment :**

TAt = TA\_STt + TA\_MTt

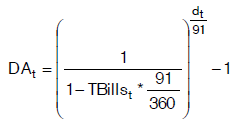




“TA\_ST\_Pct” is the Trading Adjustment applicable to ST\_Index and is equal to 0.0010

“TA\_MT\_Pct” is the Trading Adjustment applicable to MT\_Index and is equal to 0.0020

1. **Daily Accrual :**



“TBillst”, for any Index Calculation Day, t, is the 3-month T-Bill rate reported by Bloomberg, or its successor (the “USB3MTA Index”) on such Index Calculation Day, t; *provided*, that if no number is published for the USB3MTA Index rate by Bloomberg, or if the most recent such rate reported by Bloomberg is more than three (3) weeks old, then “TBillst” shall be the rate reported by the United States Treasury Department on its official website as the “Investment Rate” under “Auction Results” for 13-week Bills.

“dt” is the number of calendar days from and excluding the current Index Calculation Date, t, until and including the following Index Calculation Day, t+1

1. **Cash Units :**
2. **Cash Open Units** :

Index Level(t) – Sumproduct(Constituent Open Units(t),Settlement Prices(t))

1. **Cash Close Units** :

Index Level(t) – Sumproduct(Constituent Close Units(t),Settlement Prices(t))

## Index – Introduction

**Name** : Euro Dividend Risk Factor Index - EUR

**Type** : Futures

**Index VTId** : Ei:EU:DT:XX:X:XX:SEDE:XXXX:DB

**Index Sponsor** : Deutsche Bank

**Index Base Date** : 19 December 2008

**Index Base Currency** : Euro

### Input Files

### Input Files for Composition Calculation:

1. C\_VTIndexSpecificData\_<VTId>\_<RunDate>
2. Required Fields
   1. PriceDate – Text convertible to date
   2. Fee - Float
   3. IndexLevel – Float
   4. IndexCalculationDates - Text convertible to date
   5. DailyUnitChange – Float
   6. BusinessDay - Float
3. C\_VTOpenComposition\_<VTId>\_<RunDate>
4. Required Fields
   1. GenericTicker – Text
   2. InstrumentVTId - Text
   3. Units – Float
   4. Date - Text convertible to date
5. VTCloseComposition\_<VTId>\_<RunDate>
6. Required Fields
   1. GenericTicker – Text
   2. Date - Text convertible to date
   3. InstrumentVTId - Text
   4. Units – Float
7. VTHolidayCalendar\_<VTId>\_<RunDate>
8. Required Fields
   1. Date - Text convertible to date
9. C\_VTDailyPrices\_<VTId>\_<RunDate>
10. Required Fields
    1. PriceDate – Text convertible to date
    2. GenericTicker – Text
    3. SettlementDate - Text convertible to date
    4. SpecificInstrumentVTId - Text
    5. Price - Float

**Input Files for Level Calculation:**

1. C\_CloseCompositionForLevel\_<VTId>\_<RunDate>
   * 1. Required Fields
        1. Units (Float)
        2. Price (Float)
        3. FXRate (Float)
        4. PriceFactor (Float)
        5. FxFactor (Float)
     2. SumProduct of the above mentioned Fields gives the Index Level which is published in P\_IndexLevel\_<VTID\_Date> File

### Output Files

#### Output Files for Composition Calculation:

1. P\_VTCloseComposition\_<VTId>\_<RunDate>
2. Required Fields
   * + 1. GenericTicker – Text
       2. Date - Text convertible to date
       3. InstrumentVTId - Text
       4. Units – Float
3. P\_VTOpenComposition\_<VTId>\_<RunDate>
4. Required Fields
   * + 1. GenericTicker – Text
       2. InstrumentVTId - Text
       3. Units – Float
       4. Date - Text convertible to date
5. P\_VTIndexSpecificData\_<VTId>\_<RunDate>
6. Required Fields
   * + 1. PriceDate – Text convertible to date
       2. Fee - Float
       3. IndexLevel – Float
       4. IndexCalculationDates - Text convertible to date
       5. DailyUnitChange – Float
       6. BusinessDay - Float

**Output Files for Level Calculation:**

1. P\_IndexLevel\_<VTId>\_<RunDate>
   * 1. Required Fields
        1. Date (Text convertible to Date)
        2. IndexVTId (Text)
        3. IndexLevel (Float)

### Class – Python

**EDEIndexCalculation** : This is the VT class under which the calculations for

DBCUSEDE are coded

1. Required parameters for the Initialization of the class
2. **RunDate** : Date for which the Index level is calculated (from C# argument)
3. **LastRunDate** : Date previous to the Run Date (from C# argument)
4. **DailyPrices** : Data collected from the input file C\_VTDailyPrices\_<VTId>\_<RunDate>
5. **IndexSpecificData** : Data collected from the input file C\_VTOpenComposition\_<VTId>\_<RunDate>
6. **OpenComposition** : Data collected from the input file C\_VTOpenComposition\_<VTId>\_<RunDate>
7. **CloseComposition** : Data collected from the input file C\_VTCloseComposition\_<VTId>\_<RunDate>
8. **HolidayCalendar** : Data collected from the input file C\_VTHolidayCalendar\_<VTId>\_<RunDate>
9. **NewCloseComposition** : Dataframe inititated similar to OpenComposition
10. **OpenIndexSpecificData** : Dataframe inititated similar to IndexSpecificData
11. **NewOpenComposition** : Dataframe inititated similar to OpenComposition
12. **RebalFlag** : (=0) Intiated for Rebalance check
13. **CloseCost :** (=0) Intiated for cost calculation
14. **CashUnits :** (=0) Intiated for cost calculation
15. **IndexLevel :** (=0) Intially set to 0
16. **BusinessDays :** (=0) Intially set to 0
17. **directory** : Directory where the input and output files are stored
18. **IndexVTId** : VTId of the Index (from C# argument)
19. **log** : plogger is used for logging the processes
20. According to the Master Mapping there is one sub-process called for this **VtCompOpenClose** which calls two functions from the VT class, **fn\_VTCloseComp** and **fn\_VTOpenComp**
21. **fn\_VTCloseComp** : The output file **P\_VTCloseComposition\_<VTId>\_<RunDate>** is generated here. The following methods are called
22. **fn\_EDERebalCheck :** Takes input from Daily Prices the Price Date and the Settlement Date of the first nearest maturity contract and checks for Rebalance
23. **fn\_EDECalculateCloseComposition**: Here the **OpenComposition** dataframe is copied to **NewCloseComposition** without the cash units for close composition and date is changed to RunDate
24. **fn\_EDECalculateCloseCost**: This method takes input data from the files C\_VTIndexSpecificData\_<VTId>\_<RunDate> and **NewCloseComposition.** Here the close cost is calculated on the basis of Business logic mentioned in Rule book. Explained in Business Logic section
25. **fn\_EDECalculateCashUnits** : This method takes input data from the files C\_VTDailyPrices\_<VTId>\_<RunDate> and C\_VTIndexSpecificData\_<VTId>\_<RunDate> and also uses **NewCloseComposition** and **OpenComposition.** Here the calculation of Close Cash Units is done. Explained in Business Logic section
26. **fn\_VTOpenComp** : The output files **P\_VTOpenComposition\_<VTId>\_<RunDate>** and **P\_VTIndexCalculationDetails\_<VTId>\_<RunDate>** are generated here. This function calls for two more methods
27. **fn\_EDECalculateIndexLevel** : This method takes data from C\_VTDailyPrices\_<VTId>\_<RunDate> and **NewCloseComposition** for the Index level calculation. Explained in Business logic
28. **fn\_EDECalculateIndexSpecificDetails** : This method takes input data from C\_VTIndexSpecificData\_<VTId>\_<RunDate>. Makes a copy of it and makes open index specific data according to the Rebal flag. Here the Index Specific Data is modified as per the instructions of Rulebook. The Weights, Units, Signal, Cost are calculated for the Open Date. Explained in the Business Logic Section
29. **fn\_EDECalculateOpenComposition** : This method takes input data from the files C\_VTDailyPrices\_<VTId>\_<RunDate> and C\_VTIndexSpecificData\_<VTId>\_<RunDate> and also uses **NewCloseComposition** and **OpenComposition.** According to the Rebal Flag here the details of of index specific data are changed. The **OpenComposition** dataframe is copied to **NewOpenComposition** for open composition and date is changed to RunDate. Open cash units are also calculated and replaced here

**Index Level File**

1. **fn\_StdComp2Level :** Generates P\_IndexLevel\_<VTID\_Date> using the Input File

C\_ CloseCompositionForLevel\_<VTID\_Date>.

Index Level is the sumproduct of the following fields :

1. Units
2. Price
3. FXRate
4. PriceFactor
5. FxFactor

### Business Logic – Index Calculation :

This Index is rebalanced everyday

1. **Calculation of the Excess Return Index Level** :



ERIL(t) = the Excess Return Index Level in respect of Index Calculation Date(t)

ERIL(t-1) = the Excess Return Index Level in respect of the first Undisrupted Index Calculation Date immediately preceding Index Calculation Date(t), or, initially, the Index Commencement Date

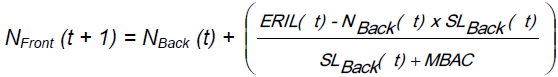
Ni(t) = Number of Units for Index Constituent(i) determined in respect of Index Calculation Date(t)

SLi(t) = the Settlement Level of Index Constituent(i) in respect of Index Calculation Date(t)

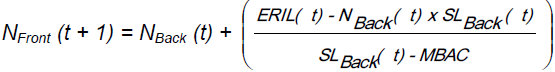
SLi(t-1) = the Settlement Level of Index Constituent(i) in respect of the first Undisrupted Index Calculation Date immediately preceding Index Calculation Date(t) or, initially, the Index Commencement Date

Cost(t) = the Cost in respect of Index Calculation Date(t) which shall be calculated as below

1. **Determination of Front Futures Contract and Back Futures Contract:** 
   * + - 1. the Front Futures Contract determined in respect of the Index Reconstitution Date immediately preceding Index Reconstitution Date(tr) (or, if none, the Index Commencement Date) shall cease to be an Index Constituent
         2. the Back Futures Contract determined in respect of the Index Reconstitution Date immediately preceding Index Reconstitution Date(tr) (or, if none, the Index Commencement Date) shall be the front futures contract in respect of Index Reconstitution Date(tr) (the "**Front Futures Contract**")
         3. the Futures Contract with an Expiry Date falling in December of the year falling two years after the year in which Index Reconstitution Date(tr) falls shall be the back futures contract in respect of Index Reconstitution Date(tr) with an initial Number of Units as of Index Reconstitution Date(tr) equal to zero (the "**Back Futures Contract**")
2. **Number of Units for the Front Futures Contract on an Index Reconstitution Date :**
   * + - 1. If *ERIL*(t) *– NBack*(t) *x SLBack*(t)is greater than zero:



* + - * 1. If (a) above does not apply then :



NBack(t) = the Number of Units in respect of Index Reconstitution Date(tr)

NFront(t+1) = the Number of Units for the Front Futures Contract in respect of the Index Calculation Date immediately following Index Reconstitution Date(tr)

SLBack(t) = the Settlement Level in respect of Index Reconstitution Date(tr)

1. **Determination of the Reconstitution Cost :**



Cost (t + 1) = the Cost in respect of the Index Calculation Date(t+1) immediately following Index Reconstitution Date(tr)

1. **Calculation of the Daily Unit Change :**
   * + - 1. Equals the Number of Units in respect of Index Reconstitution Date(tr) of the Front Futures Contract determined in respect of Index Reconstitution Date(tr)
         2. Equals the number of Index Calculation Dates falling in the period from (and including) Index Reconstitution Date(tr) to (but excluding) the next following Index Reconstitution Date
         3. The Daily Unit Change as of the Index Commencement Date is equal to 0.039289643
2. **Number of Units for the Front Futures Contract :**

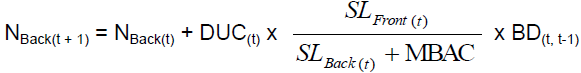
Ni(t+1) = Ni(t)

Ni(t+1) = the Number of Units in respect of Front Futures Contract(i) and the Index Calculation Date(t+1) immediately following Index Calculation Date(t)

Ni(t) = the Number of Units in respect of Front Futures Contract(i) and Index Calculation Date(t)

As of the Index Commencement Date the Number of Units of the Front Futures Contract is 9.900990099

1. **Number of Units for the Back Futures Contract :**



DUC(t) = the Daily Unit Change determined in respect of the Index Reconstitution Date, or if none the Index Commencement Date immediately preceding or coinciding with such Index Calculation Date(t);

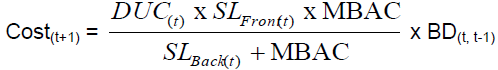
NBack(t+1) = the Number of Units of the Back Futures Contract in respect of the Index Calculation Date(t+1) immediately following Index Calculation Date(t);

SLFront(t) = the Settlement Level of the Front Futures Contract in respect of Index CalculationDate(t)

BD(t, t-1) = the number of Index Calculation Dates in the period from but excluding the first Undisrupted Index Calculation Date immediately preceding Index Calculation Date(t), to and including Index Calculation Date(t).

As of the Index Commencement Date the Number of Units of the Back Futures Contract is zero

1. **Calculation of the Daily Cost:**



1. **Cash Units :**
2. **Cash Open Units** :

Index Level(t) – Sumproduct(Constituent Open Units(t),Settlement Prices(t))

1. **Cash Close Units** :

Index Level(t) – Sumproduct(Constituent Close Units(t),Settlement Prices(t))

## Index – Introduction

**Name** : Euro Dividend USD Hedged Index

**Type** : Futures

**Index VTId** : Ei:US:DT:XX:X:XX:SEDU:XXXX:DB

**Index Sponsor** : Deutsche Bank

**Index Base Date** : 19 December 2008

**Index Base Currency** : US Dollar

### Input Files

### Input Files for Composition Calculation:

1. C\_VTIndexSpecificData\_<VTId>\_<RunDate>
2. Required Fields
   1. PriceDate – Text convertible to date
   2. Units - Float
   3. RebalancingTransactionCost - Float
3. C\_VTOpenComposition\_<VTId>\_<RunDate>
4. Required Fields
   1. GenericTicker – Text
5. C\_VTDailyPrices\_<VTId>\_<RunDate>
6. Required Fields
   1. PriceDate – Text convertible to date
   2. OpenDate - Text convertible to date
   3. Price - Float
   4. IndexLevel - Float
   5. FxRate - Float
   6. Units – Float
7. C\_VTUnderlyingCloseComposition\_<VTId>\_<RunDate>
8. Required Fields
   1. Units – Float
   2. Price – Float
9. C\_VTUnderlyingOpenComposition\_<VTId>\_<RunDate>
10. Required Fields
    1. Units – Float
    2. Price – Float

**Input Files for Level Calculation:**

1. C\_CloseCompositionForLevel\_<VTId>\_<RunDate>
   * 1. Required Fields
        1. Units (Float)
        2. Price (Float)
        3. FXRate (Float)
        4. PriceFactor (Float)
        5. FxFactor (Float)
     2. SumProduct of the above mentioned Fields gives the Index Level which is published in P\_IndexLevel\_<VTID\_Date> File

### Output Files

#### Output Files for Composition Calculation:

1. P\_VTCloseComposition\_<VTId>\_<RunDate>
2. Required Fields
   * + 1. GenericTicker – Text
       2. Date - Text convertible to date
       3. InstrumentVTId - Text
       4. Units – Float
3. P\_VTOpenComposition\_<VTId>\_<RunDate>
4. Required Fields
   * + 1. GenericTicker – Text
       2. InstrumentVTId - Text
       3. Units – Float
       4. Date - Text convertible to date
5. P\_VTIndexSpecificData\_<VTId>\_<RunDate>
6. Required Fields
   * + 1. PriceDate – Text convertible to date
       2. Units – Float

**Output Files for Level Calculation:**

1. P\_IndexLevel\_<VTId>\_<RunDate>
   * 1. Required Fields
        1. Date (Text convertible to Date)
        2. IndexVTId (Text)
        3. IndexLevel (Float)

### Class – Python

**EDUIndexCalculation**: This is the VT class under which the calculations for

DBCUSEDU are coded

1. Required parameters for the Initialization of the class
2. **RunDate** : Date for which the Index level is calculated (from C# argument)
3. **LastRunDate** : Date previous to the Run Date (from C# argument)
4. **OpenDate** : The next trading date of the index. Taken from Daily Prices file
5. **DailyPrices** : Data collected from the input file C\_VTDailyPrices\_<VTId>\_<RunDate>
6. **IndexSpecificData** : Data collected from the input file C\_VTOpenComposition\_<VTId>\_<RunDate>
7. **OpenComposition** : Data collected from the input file C\_VTOpenComposition\_<VTId>\_<RunDate>
8. **UnderlyingCloseComposition**: Data collected from the input file C\_VTUnderlyingCloseComposition\_<VTId>\_<RunDate>
9. **UnderlyingOpenComposition** : Data collected from the input file C\_VTUnderlyingOpenComposition\_<VTId>\_<RunDate>
10. **NewCloseComposition** : Dataframe inititated similar to OpenComposition
11. **OpenIndexSpecificData** : Dataframe inititated similar to IndexSpecificData
12. **NewOpenComposition** : Dataframe inititated similar to OpenComposition
13. **RebalFlag** : (=0) Intiated for Rebalance check
14. **IndexLevel :** (=0) Intially set to 0
15. **directory** : Directory where the input and output files are stored
16. **IndexVTId** : VTId of the Index (from C# argument)
17. **log** : plogger is used for logging the processes
18. According to the Master Mapping there is one sub-process called for this **VtCompOpenClose** which calls two functions from the VT class, **fn\_VTCloseComp** and **fn\_VTOpenComp**
19. **fn\_VTCloseComp** : The output file **P\_VTCloseComposition\_<VTId>\_<RunDate>** is generated here. The following methods are called
20. **fn\_EDURebalCheck :** Takes input the RunDate fetched from C# argument and checks for Rebalance
21. **fn\_EDUCloseComposition**: This takes data for units from UnderlyingCloseComposition, OpenComposition and IndexSpecificData and the price data from DailyPrices. Here the **NewCloseComposition** along with the cash units for close composition are calculated
22. **fn\_VTOpenComp** : The output files **P\_VTOpenComposition\_<VTId>\_<RunDate>** and **P\_VTIndexCalculationDetails\_<VTId>\_<RunDate>** are generated here. This function calls for two more methods
23. **fn\_EDUCalculateIndexLevel** : This method takes data from **NewCloseComposition,** UnderlyingCloseComposition and DailyPricesfor the Index level calculation. Explained in Business logic
24. **fn\_EDUOpenIndexSpecificDetails** : This method takes input data from C\_VTIndexSpecificData\_<VTId>\_<RunDate>. Makes a copy of it and makes open index specific data according to the Rebal flag. Here the Index Specific Data is modified as per the instructions of Rulebook. The Units, RebalancingTransactionCost are calculated for the Open Date. Explained in the Business Logic Section
25. **fn\_EDUOpenComposition** : This method takes input data from the files C\_VTDailyPrices\_<VTId>\_<RunDate> and C\_VTIndexSpecificData\_<VTId>\_<RunDate> and also uses UnderlyingOpenCompositionNewCloseCompositionand OpenComposition**.** According to the Rebal Flag here the details of of the **OpenComposition** dataframe are calculated and are copied to **NewOpenComposition** for open composition. Open cash units are also calculated and replaced here

**Index Level File**

1. **fn\_StdComp2Level :** Generates P\_IndexLevel\_<VTID\_Date> using the Input File

C\_ CloseCompositionForLevel\_<VTID\_Date>.

Index Level is the sumproduct of the following fields :

1. Units
2. Price
3. FXRate
4. PriceFactor
5. FxFactor

### Business Logic – Index Calculation :

Rebalance - the third Friday of each month in each year following the Index Commencement Date, or if any such day is not an Index Calculation Date, the immediately preceding Index Calculation Date

1. **Determination of the Index Closing Level** :



1. **Rebalancing Transaction Cost:** 
   * + - 1. if Index Calculation Date (*t*) is the Index Commencement Date:

RTC(t) = 0

* + - * 1. if Index Calculation Date (*t*) is any other Rebalancing Date:



* + - * 1. Otherwise:

RTC(t) = 0

1. **Unit Exposure:**



RTC(t-1) - the Rebalancing Transaction Cost in respect of Index Calculation Date (*t-1*)

Unit(tR) - in respect of Index Calculation Date (t), Unit(t) for Rebalancing Date (tR)

FX(t) - the Closing Exchange Rate in respect of Index Calculation Date (t)

FX(tC) - the Closing Exchange Rate in respect of Selection Date (tC)

IL(t) - Index Closing Level in respect of Index Calculation Date (t)

IL(t-1) - the Index Closing Level in respect of Index Calculation Date (t-1)

IL(tC) - the Index Closing Level in respect of Selection Date (tC) or, if Selection Date (tc) falls prior to the Index Commencement Date, the Index Base Level

TC(t) - TC(t) = 0.50% + (0.50%\*mt/12)

mt - an integer corresponding to the month of the year

UL(t) - the Underlying Index Closing Level in respect of Index Calculation Date (t)

UL(t-1) - the Underlying Index Closing Level in respect of Index Calculation Date (t-1)

UL(tC) - the Underlying Index Closing Level in respect of Selection Date (tC)

1. **Cash Units :**
2. **Cash Open Units** :

Index Level(t) – Sumproduct(Constituent Open Units(t),Settlement Prices(t))

1. **Cash Close Units** :

Index Level(t) – Sumproduct(Constituent Close Units(t),Settlement Prices(t))

## Index – Introduction

**Name** : BofAML Mean Reversion EUR Index TR

**Type** : Strategy

**Index VTId** : Ei:EU:DT:XX:X:XX:MRET:XXXX:ML

**Index Sponsor** : Merrill Lynch International

**Index Base Date** : 26 October 1994

**Index Base Currency** : EUR

### Input Files

### Input Files for Composition Calculation:

1. C\_VTIndexSpecificData\_<VTId>\_<RunDate>
2. Required Fields
   1. PriceDate – Text convertible to date
   2. SpecificTicker – Text
   3. FundingRate – Float
   4. IndexLevel – Float
   5. FundingRate – Float
   6. Fee – Float
   7. FinalExposure - Float
   8. RebalancingTransactionCost - Float
3. C\_VTOpenComposition\_<VTId>\_<RunDate>
4. Required Fields
   1. SpecificTicker – Text
   2. Units – Float
   3. Date - Text convertible to date
5. C\_VTDailyPrices\_<VTId>\_<RunDate>
6. Required Fields
   1. PriceDate – Text convertible to date
   2. GenericTicker - Text
   3. Price - Float
7. C\_VTTickPrices\_<VTId>\_<RunDate>
8. Required Fields
   1. PriceDate – Text convertible to date
   2. EURUSD\_F160 - Float
   3. EURUSD\_F153 - Float
   4. SpecificTicker - Text
   5. Dividend - Float
   6. TickPrice - Float

**Input Files for Level Calculation:**

1. C\_CloseCompositionForLevel\_<VTId>\_<RunDate>
   * 1. Required Fields
        1. Units (Float)
        2. Price (Float)
        3. FXRate (Float)
        4. PriceFactor (Float)
        5. FxFactor (Float)
     2. SumProduct of the above mentioned Fields gives the Index Level which is published in P\_IndexLevel\_<VTID\_Date> File

### Output Files

#### Output Files for Composition Calculation:

1. P\_VTCloseComposition\_<VTId>\_<RunDate>
2. Required Fields
   * + 1. SpecificTicker – Text
       2. Units – Float
       3. Date - Text convertible to date
3. P\_VTOpenComposition\_<VTId>\_<RunDate>
4. Required Fields
   * + 1. SpecificTicker – Text
       2. Units – Float
       3. Date - Text convertible to date
5. P\_VTIndexSpecificData\_<VTId>\_<RunDate>
6. Required Fields
   * + 1. PriceDate – Text convertible to date
       2. Units – Float

**Output Files for Level Calculation:**

1. P\_IndexLevel\_<VTId>\_<RunDate>
2. Required Fields
3. Date (Text convertible to Date)
4. IndexVTId (Text)
5. IndexLevel (Float)

### Class – Python

**MRETIndexCalculation**: This is the VT class under which the calculations for

MLFPMRET are coded

1. Required parameters for the Initialization of the class
2. **RunDate** : Date for which the Index level is calculated (from C# argument)
3. **LastRunDate** : Date previous to the Run Date (from C# argument)
4. **DailyPrices** : Data collected from the input file C\_VTDailyPrices\_<VTId>\_<RunDate>
5. **IndexSpecificData** : Data collected from the input file C\_VTOpenComposition\_<VTId>\_<RunDate>
6. **OpenComposition** : Data collected from the input file C\_VTOpenComposition\_<VTId>\_<RunDate>
7. **TickPrices :** Data collected from the input file C\_VTTickPrices\_<VTId>\_<RunDate>
8. **NewCloseComposition** : Dataframe inititated similar to OpenComposition
9. **OpenIndexSpecificData** : Dataframe inititated similar to IndexSpecificData
10. **NewOpenComposition** : Dataframe inititated similar to OpenComposition
11. **IndexLevel :** (=0) Intially set to 0
12. **temp\_df :** A temperory dataframe is intiated to store interim calculations
13. **directory** : Directory where the input and output files are stored
14. **IndexVTId** : VTId of the Index (from C# argument)
15. **log** : plogger is used for logging the processes
16. According to the Master Mapping there is one sub-process called for this **VtCompOpenClose** which calls two functions from the VT class, **fn\_VTCloseComp** and **fn\_VTOpenComp**
17. **fn\_VTCloseComp** : The output file **P\_VTCloseComposition\_<VTId>\_<RunDate>** is generated here. The following methods are called
18. **fn\_MRETCloseComposition**: Here the **NewCloseComposition** is created by making copy of the OpenComposition wihout the cash units
19. **fn\_MRETCalculateIndexLevel:** This method takes data from NewCloseComposition, DailyPrices, TickPrices and IndexSpecificData. Also it calls a helper function fn\_CalculateDayFraction. The Index level is calculated according to the formula mention in the Business Logic
20. **fn\_MRETCalculateCloseCashUnits:** This method takes data from NewCloseComposition, DailyPrices, TickPrices. The close Cash units are calculated here according to the business logic mentioned below
21. **fn\_VTOpenComp** : The output files **P\_VTOpenComposition\_<VTId>\_<RunDate>** and **P\_VTIndexCalculationDetails\_<VTId>\_<RunDate>** are generated here. This function calls for two more methods
22. **fn\_MRETOpenComposition** : This method takes input data from the files DailyPrices, TickPrices and NewOpenComposition. Two Helper functions are also defined fn\_Average and fn\_VolatilityInterim for the average and volatility calculations as mentioned in the Business Logic
23. **fn\_MRETOpenIndexSpecificData()** : This method takes input data from C\_VTIndexSpecificData\_<VTId>\_<RunDate>. Makes a copy of it and makes open index specific data. Here the Index Specific Data is modified as per the instructions of Rulebook. The FinalExposure, RebalancingTransactionCost are calculated. Explained in the Business Logic Section

**Index Level File**

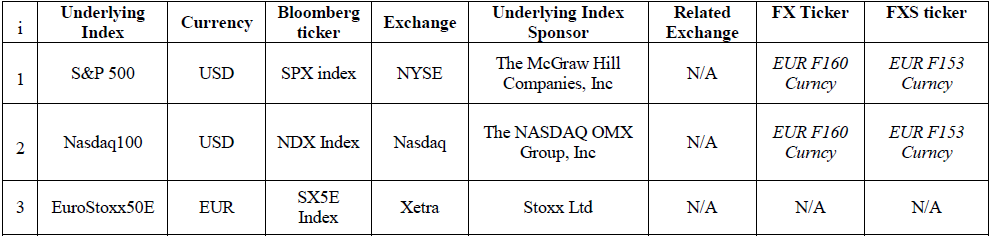
1. **fn\_StdComp2Level :** Generates P\_IndexLevel\_<VTID\_Date> using the Input File

C\_ CloseCompositionForLevel\_<VTID\_Date>.

Index Level is the sumproduct of the following fields :

1. Units
2. Price
3. FXRate
4. PriceFactor
5. FxFactor

### Business Logic – Index Calculation : Underlyings of the MLFPMRET index



Rebalance – Every Index Calculation day. Units are calculated on every Calculation day

1. **Excess Return Component Strategy Calculation** :



1. **Delta Adjustment Factorit** :

Indext-1 x Abs(Final Exposureit – Final Exposureit-1) x 0.03% / 3

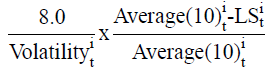
1. **Daily Number of Underlying Index Units:**



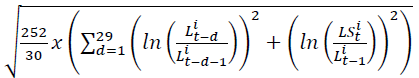
1. **Final Exposureit:**



1. **Exposureit:**



1. **Volatilityit:**



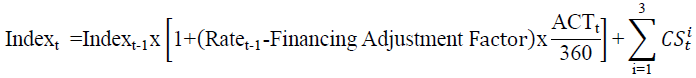
1. **Average(10)it:**



1. **Average(200)it:**



1. **Index Value Calculation :**



CSit: means the Excess Return of component strategy “i” in respect of Index Valuation Date”t”

Dividendit : the index dividend points published by Bloomberg under the Index Ticker “i”

Lit: the level of Underlying Index “i” as published by Bloomberg under the Index Ticker “i”

Lit-1 : the level of Underlying Index “i” as published by Bloomberg under the Index Ticker “i” and with respect to the immediately preceding Index Valuation Date

FXit : the Exchange Rate with respect to Currency “i”on Index Valuation Date “t”

FXit-1 : the Exchange Rate with respect to Currency “i” on the immediately preceding Index Valuation Date “t”

Ratet-1 : the Interest Rate as of the immediately preceding Index Valuation Date

ACTt-1 : the number of calendar days from (but excluding) the immediately preceding Index Valuation Date to (and including) Index Valuation Date “t”

Indicatorit-1 : Shall be equal to 1 if LSit <= Average(200)it Otherwise shall be 0

LSit : the first level of Underlying Index i as published by Bloomberg under the Index Ticker “i” during the period commencing 15 minutes prior to the official closing time of Underlying In ex “i” on Index Valuation Date “t”

FXSit : the mid foreign exchange rate used to convert one unit of Euro into units of such currency as published by Bloomberg with respect to 3.30 p.m. New York time

Indext : the Index Value on Index Valuation Date “t”

Finance Adjusting Factor : (= 0.5%)

1. **Cash Units :**
2. **Cash Open Units** :

Index Level(t) – Sumproduct(Constituent Open Units(t),Settlement Prices(t))

1. **Cash Close Units** :

Index Level(t) – Sumproduct(Constituent Close Units(t),Settlement Prices(t))

## Index – Introduction

**Name** : Citi Commodities Curve Composite (BCOMSM Weighted) Balanced Alpha Index

**Type** : Futures / Commodities

**Index VTId** : Ci:US:XX:XX:X:XX:CADB:XXXX:CI

**Index Sponsor** : Citigroup Global Markets Limited

**Index Base Date** : 31 March 1999

**Index Base Currency** : U.S. Dollars

**Underlyings**: CITI Commodity Excess Return Indices

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| i | Component Index | Component Index Ticker | Associated Commodity | Exchange |
| 1 | CUBES-D WTI Crude Oil (ER) | CCUDCLER <Index> | WTI Crude Oil | NYMEX |
| 2 | CUBES-D Brent Crude Oil (ER) | CCUDCOER <Index> | Brent Crude Oil | ICE |
| 3 | CUBES-D Gasoline (ER) | CCUDXBER <Index> | Gasoline | NYMEX |
| 4 | CUBES-D Heating Oil (ER) | CCUDHOER <Index> | Heating Oil | NYMEX |
| 5 | Contract Momentum Natural Gas (ER) | CVICCMNG <Index> | Natural Gas | NYMEX |
| 6 | CUBES-D Aluminium (ER) | CCUDLAER <Index> | Aluminium | LME |
| 7 | CUBES-D Copper (CMX) (ER) | CCUDHGER <Index> | Copper (CMX) | COMEX |
| 8 | CUBES-D Nickel (ER) | CCUDLNER <Index> | Nickel | LME |
| 9 | CUBES-D Zinc (ER) | CCUDLXER <Index> | Zinc | LME |
| 10 | CUBES-D Gold (ER) | CCUDGCER <Index> | Gold | COMEX |
| 11 | CUBES-D Silver (ER) | CCUDSIER <Index> | Silver | COMEX |
| 12 | Contract Momentum Wheat (CBOT) (ER) | CVICCMWX <Index> | Wheat (CBOT) | CBOT |
| 13 | Contract Momentum Wheat (KCBOT) (ER) | CVICCMKW <Index> | Wheat (KCBOT) | KCBOT |
| 14 | Contract Momentum Corn (ER) | CVICCMCX <Index> | Corn | CBOT |
| 15 | Contract Momentum Soybeans (ER) | CVICCMSX <Index> | Soybean | CBOT |
| 16 | CUBES-D Soybean Oil (ER) | CCUDBOER <Index> | Soybean Oil | CBOT |
| 17 | CUBES-D Soybean Meal (ER) | CCUDSMER <Index> | Soybean Meal | CBOT |
| 18 | CUBES-D Cotton (ER) | CCUDCTER <Index> | Cotton | NYBOT |
| 19 | CUBES-D Sugar (ER) | CCUDSBER <Index> | Sugar | NYBOT |
| 20 | CUBES-D Coffee (ER) | CCUDKCER <Index> | Coffee | NYBOT |
| 21 | CUBES-D Cocoa (ER) | CCUDCCER <Index> | Cocoa | NYBOT |
| 22 | CUBES-D Live Cattle (ER) | CCUDLCER <Index> | Live Cattle | CME |
| 23 | CUBES-D Lean Hogs (ER) | CCUDLHER <Index> | Lean Hogs | CME |

**Underlyings**: Bloomberg Benchmark Indices

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | Bloomberg WTI Crude Oil Subindex | BCOMCL <Index> | WTI Crude Oil | NYMEX |
| 2 | Bloomberg Brent Crude Oil Subindex | BCOMCO <Index> | Brent Crude Oil | ICE |
| 3 | Bloomberg Unleaded Gasoline Subindex | BCOMRB <Index> | Gasoline | NYMEX |
| 4 | Bloomberg Heating Oil Subindex | BCOMHO <Index> | Heating Oil | NYMEX |
| 5 | Bloomberg Natural Gas Subindex | BCOMNG <Index> | Natural Gas | NYMEX |
| 6 | Bloomberg Aluminum Subindex | BCOMAL <Index> | Aluminium | LME |
| 7 | Bloomberg Copper Subindex | BCOMHG <Index> | Copper (CMX) | COMEX |
| 8 | Bloomberg Nickel Subindex | BCOMNI <Index> | Nickel | LME |
| 9 | Bloomberg Zinc Subindex | BCOMZS <Index> | Zinc | LME |
| 10 | Bloomberg Gold Subindex | BCOMGC <Index> | Gold | COMEX |
| 11 | Bloomberg Silver Subindex | BCOMSI <Index> | Silver | COMEX |
| 12 | Bloomberg Wheat Subindex | BCOMWH <Index> | Wheat (CBOT) | CBOT |
| 13 | Bloomberg Kansas Wheat Subindex | BCOMKW <Index> | Wheat (KCBOT) | KCBOT |
| 14 | Bloomberg Corn Subindex | BCOMCN <Index> | Corn | CBOT |
| 15 | Bloomberg Soybeans Subindex | BCOMSY <Index> | Soybean | CBOT |
| 16 | Bloomberg Soybean Oil Subindex | BCOMBO <Index> | Soybean Oil | CBOT |
| 17 | Bloomberg Soybean Meal Subindex | BCOMSM <Index> | Soybean Meal | CBOT |
| 18 | Bloomberg Cotton Subindex | BCOMCT <Index> | Cotton | NYBOT |
| 19 | Bloomberg Sugar Subindex | BCOMSB <Index> | Sugar | NYBOT |
| 20 | Bloomberg Coffee Subindex | BCOMKC <Index> | Coffee | NYBOT |
| 21 | Bloomberg Cocoa Subindex | BCOMCC <Index> | Cocoa | NYBOT |
| 22 | Bloomberg Live Cattle Subindex | BCOMLC <Index> | Live Cattle | CME |
| 23 | Bloomberg Lean Hogs Subindex | BCOMLH <Index> | Lean Hogs | CME |

### Input Files

### Input Files for Composition Calculation:

1. C\_VTIndexSpecificData\_<VTId>\_<RunDate>
2. Required Fields
   1. PriceDate – Text convertible to date
   2. SpecificTicker – Text
   3. FundingRate – Float
   4. IndexLevel – Float
   5. FundingRate – Float
   6. Fee – Float
   7. FinalExposure - Float
   8. RebalancingTransactionCost - Float
3. C\_VTOpenComposition\_<VTId>\_<RunDate>
4. Required Fields
   1. SpecificTicker – Text
   2. Units – Float
   3. Date - Text convertible to date
5. C\_VTDailyPrices\_<VTId>\_<RunDate>
6. Required Fields
   1. PriceDate – Text convertible to date
   2. GenericTicker - Text
   3. Price - Float
7. C\_VTCloseComposition\_<VTId>\_<RunDate>
8. Required Fields
   1. PriceDate – Text convertible to date
   2. SpecificTicker – Text
   3. Units – Float
   4. Date – Text convertible to date
9. C\_VTPriorRollPrices\_<VTId>\_<RunDate>
10. Required Fields
    1. PriceDate – Text convertible to date
    2. SpecificInstrumentVTId - Text
11. C\_VTBCOMWeights\_<VTId>\_<RunDate>
12. Required Fields
    1. Date – Text convertible to date
    2. AssociatedCommodity - Text
13. C\_VTHolidayCalendar\_<VTId>\_<RunDate>
14. Required Fields
    1. Date – Text convertible to date

**Input Files for Level Calculation:**

1. C\_CloseCompositionForLevel\_<VTId>\_<RunDate>
   * 1. Required Fields
        1. Units (Float)
        2. Price (Float)
        3. FXRate (Float)
        4. PriceFactor (Float)
        5. FxFactor (Float)
     2. SumProduct of the above mentioned Fields gives the Index Level which is published in P\_IndexLevel\_<VTID\_Date> File

### Output Files

#### Output Files for Composition Calculation:

1. P\_VTCloseComposition\_<VTId>\_<RunDate>
2. Required Fields
   * + 1. SpecificTicker – Text
       2. Units – Float
       3. Date - Text convertible to date
3. P\_VTOpenComposition\_<VTId>\_<RunDate>
4. Required Fields
   * + 1. SpecificTicker – Text
       2. Units – Float
       3. Date - Text convertible to date
5. P\_VTIndexSpecificData\_<VTId>\_<RunDate>
6. Required Fields
   * + 1. PriceDate – Text convertible to date
       2. Units – Float

**Output Files for Level Calculation:**

1. P\_IndexLevel\_<VTId>\_<RunDate>
2. Required Fields
3. Date (Text convertible to Date)
4. IndexVTId (Text)
5. IndexLevel (Float)

### Class – Python

**CitiBCOMIndexCalculation**: This is the VT class under which the calculations for

CVICCADB are coded

1. Required parameters for the Initialization of the class
2. **RunDate** : Date for which the Index level is calculated (from C# argument)
3. **LastRunDate** : Date previous to the Run Date (from C# argument)
4. **OpenDate** :The next trading date fetched from Daily Prices file
5. **DailyPrices** : Data collected from the input file C\_VTDailyPrices\_<VTId>\_<RunDate>
6. **IndexSpecificData** : Data collected from the input file C\_VTOpenComposition\_<VTId>\_<RunDate>
7. **CloseComposition** : Data collected from the input file C\_CloseComposition\_<VTId>\_<RunDate>
8. **OpenComposition** : Data collected from the input file C\_VTOpenComposition\_<VTId>\_<RunDate>
9. **BCOMWeights :** Data collected from the input file C\_BCOMWeights\_<VTId>\_<RunDate>
10. **PriorRollPrices :** Data collected from the input file C\_PriorRollPrices\_<VTId>\_<RunDate>
11. **HolidayCalendar :** Data collected from the input file C\_HolidayCalendar\_<VTId>\_<RunDate>
12. **BCOMWeights :** Data collected from the input file C\_BCOMWeights\_<VTId>\_<RunDate>
13. **NewCloseComposition** : Dataframe inititated similar to OpenComposition
14. **OpenIndexSpecificData** : Dataframe inititated similar to IndexSpecificData
15. **NewOpenComposition** : Dataframe inititated similar to OpenComposition
16. **IndexLevel :** (=0) Intially set to 0
17. **temp\_df :** A temperory dataframe is intiated to store interim calculations
18. **directory** : Directory where the input and output files are stored
19. **IndexVTId** : VTId of the Index (from C# argument)
20. **log** : plogger is used for logging the processes
21. According to the Master Mapping there is one sub-process called for this **VtCompOpenClose** which calls two functions from the VT class, **fn\_VTCloseComp** and **fn\_VTOpenComp**
22. **fn\_VTCloseComp** : The output file **P\_VTCloseComposition\_<VTId>\_<RunDate>** is generated here. The following methods are called
23. **fn\_CitiBCOMRebalCheck** : Here a check is applied for Rebal Date. Rebalance happens on every first working day of the month
24. **fn\_CitiBCOMCloseComposition :** Here the OpenComposition is copied as the CloseComposition for the next day except for Cash Units. The Cash Units are calculated using the Index Level for Run Date
25. **fn\_VTOpenComp** : The output files **P\_VTOpenComposition\_<VTId>\_<RunDate>** and **P\_VTIndexCalculationDetails\_<VTId>\_<RunDate>** are generated here. This function calls for two more methods
26. **fn\_CitiBCOMOpenComposition** : This method takes input data from the files DailyPrices, PriorRoll Prices and NewOpenComposition. This method also calls for fn\_CitiBCOMRebalComposition, which calculates Rebal composition on every first working day of the month. If there is a rebal the RebalComposition becomes the Open Composition except for Open Cash. Else the NewOpenComposition is a copy of the OpenComposition except for cash units.
27. **fn\_CitiBCOMOpenIndexSpecificData** : This method takes input data from C\_VTIndexSpecificData\_<VTId>\_<RunDate>. Here the Index Specific Data is modified as per the instructions of Rulebook. The Index Level is replaced with the new Index Level

**Index Level File**

1. **fn\_StdComp2Level :** Generates P\_IndexLevel\_<VTID\_Date> using the Input File

C\_ CloseCompositionForLevel\_<VTID\_Date>.

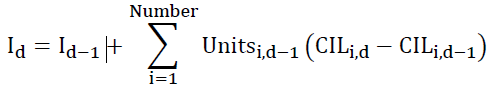
Index Level is the sumproduct of the following fields :

1. Units
2. Price
3. FXRate
4. PriceFactor
5. FxFactor

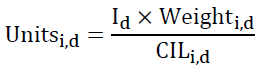
### Business Logic – Index Calculation :

Rebalance – First working day of every month.

1. **The Index Level on each Index Business Day following the Index Start Date** :



1. **Determination of the Number of Units** :



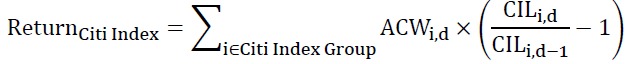
1. **Determination of the Weight of Component Index :**



1. **Beta Multiplier:**



1. **Weighted Daily Return and Lookback Period :**





|  |  |
| --- | --- |
| “Id” | shall mean the Index Level on d. |
| “Id-1” | shall mean the Index Level on the Index Business Day immediately preceding d. |
| “Number” | shall mean the number of Component Indices contained in the Index. |
| “Unitsi,d-1” | shall mean the Number of Units in respect of Component Index i on the Index Business Day immediately preceding d. |
| “CILi,d” | shall mean the Component Index Level of Component Index i on d. |
| “CILi,d-1” | shall mean the Component Index Level of Component Index i on the Index Business Day immediately preceding d. |
| “Unitsi,d” | shall mean the Number of Units in respect of i on d. |
| “Weighti,d” | shall mean the Weight of i on d. |
| “ACWi,d” | shall mean the Associated Commodity Weight of i on d. |
| “BMi,d” | shall mean the Beta Multiplier of i on d. |
| “Min” | shall mean the lesser of the two amounts in the following parentheses and separated by a comma. |
| “Max” | shall mean the greater of the two amounts in the following parentheses and separated by a comma. |
| “𝛃𝐝” | shall mean the Realized Beta for the calendar month during which d occurs. |
| “ReturnCiti Index” | shall mean the Weighted Daily Return of the Citi Index Group on d. |
| “i∈Citi Index Group” | defines “i” as each Component Index contained in the Citi Index Group. |
| “ReturnBenchmark” | shall mean the Weighted Daily Return of the Benchmark Group on d. |
| “i∈Benchmark Group” | defines “i” as each Component Index contained in the Benchmark Group. |

1. **Cash Units :**
2. **Cash Open Units** :

Index Level(t) – Sumproduct(Constituent Open Units(t),Settlement Prices(t))

1. **Cash Close Units** :

Index Level(t) – Sumproduct(Constituent Close Units(t),Settlement Prices(t))

## Index – Introduction

**Name** : BofAML Mean Reversion USD Index ER Index

**Type** : Strategies

**Index VTId** : Ei:US:DT:XX:X:XX:MRUE:XXXX:ML

**Index Sponsor** : BofAML

**Index Base Date** : 26 October 1994

**Index Base Currency** : U.S. Dollars

### Input Files

### Input Files for Composition Calculation:

1. C\_VTIndexSpecificData\_<VTId>\_<RunDate>
2. Required Fields
   1. PriceDate – Text convertible to date
   2. SpecificTicker – Text
   3. FundingRate – Float
   4. IndexLevel – Float
   5. FundingRate – Float
   6. Fee – Float
   7. FinalExposure - Float
   8. RebalancingTransactionCost - Float
3. C\_VTOpenComposition\_<VTId>\_<RunDate>
4. Required Fields
   1. SpecificTicker – Text
   2. Units – Float
   3. Date - Text convertible to date
5. C\_VTDailyPrices\_<VTId>\_<RunDate>
6. Required Fields
   1. PriceDate – Text convertible to date
   2. GenericTicker - Text
   3. Price - Float
7. C\_VTUnderlyingCloseComposition \_<VTId>\_<RunDate>
8. Required Fields
   1. SpecificTicker – Text
   2. Units – Float
   3. Date - Text convertible to date
9. C\_ VTUnderlyingOpenComposition \_<VTId>\_<RunDate>
10. Required Fields
    1. SpecificTicker – Text
    2. Units – Float
    3. Date - Text convertible to date

**Input Files for Level Calculation:**

1. C\_CloseCompositionForLevel\_<VTId>\_<RunDate>
   * 1. Required Fields
        1. Units (Float)
        2. Price (Float)
        3. FXRate (Float)
        4. PriceFactor (Float)
        5. FxFactor (Float)
     2. SumProduct of the above mentioned Fields gives the Index Level which is published in P\_IndexLevel\_<VTID\_Date> File

### Output Files

#### Output Files for Composition Calculation:

1. P\_VTCloseComposition\_<VTId>\_<RunDate>
2. Required Fields
   * + 1. SpecificTicker – Text
       2. Units – Float
       3. Date - Text convertible to date
3. P\_VTOpenComposition\_<VTId>\_<RunDate>
4. Required Fields
   * + 1. SpecificTicker – Text
       2. Units – Float
       3. Date - Text convertible to date
5. P\_VTIndexSpecificData\_<VTId>\_<RunDate>
6. Required Fields
   * + 1. PriceDate – Text convertible to date
       2. Units – Float

**Output Files for Level Calculation:**

1. P\_IndexLevel\_<VTId>\_<RunDate>
2. Required Fields
3. Date (Text convertible to Date)
4. IndexVTId (Text)
5. IndexLevel (Float)

### Class – Python

**CitiBCOMIndexCalculation**: This is the VT class under which the calculations for

MLFPMRUE are coded

1. Required parameters for the Initialization of the class
2. **RunDate** : Date for which the Index level is calculated (from C# argument)
3. **LastRunDate** : Date previous to the Run Date (from C# argument)
4. **OpenDate** :The next trading date fetched from Daily Prices file
5. **DailyPrices** : Data collected from the input file C\_VTDailyPrices\_<VTId>\_<RunDate>
6. **IndexSpecificData** : Data collected from the input file C\_VTOpenComposition\_<VTId>\_<RunDate>
7. **ChildOpenComp** : Data collected from the input file C\_CloseComposition\_<VTId>\_<RunDate>
8. **ChildCloseComp** : Data collected from the input file C\_VTOpenComposition\_<VTId>\_<RunDate>
9. **IndexOpenComp :** Data collected from the input file C\_BCOMWeights\_<VTId>\_<RunDate>
10. **data\_for\_RunDate** : This fetches data for the RunDate from the function data\_for\_date
11. **data\_for\_LastRunDate**: This fetches data for the LastRunDate from the function data\_for\_date
12. **child\_FX** : This stores the FX Rate used in MRET calculation i.e., F160
13. According to the Master Mapping there is one sub-process called for this **VtCompOpenClose** which calls two functions from the VT class, **fn\_VTCloseComp** and **fn\_VTOpenComp**
14. **fn\_VTCloseComp** : The output file **P\_VTCloseComposition\_<VTId>\_<RunDate>** is generated here. The following methods are called
15. **fn\_VTOpenComp** : The output files **P\_VTOpenComposition\_<VTId>\_<RunDate>** and **P\_VTIndexSpecificData\_<VTId>\_<RunDate>** are generated here. This function calls for two more methods
16. **data\_for\_date :** This function fetches relevant intrinsic data for the Index Level and composition calculation such as funding\_rate, IndexLevel(LastRunDate), EURrate, ChildLevel(MRET Level), funding\_days, ChildAdjFac
17. **OpenClose\_data :** Based on the state (Open or Close) the compostion(Units for underlying and cash) are calculated here
18. **IndexLevel\_RunDate :** Index Level for the Run Date is calculated by using the NewCloseCompostion method defined. Index Level is the sumproduct of Units, Prices and FxRates.
19. **FXAdjFactor :** As the Fx Rate used for the underlying level calculation and Parent Index Level Calculation is different. We need to take into account of the factor
20. **NewCloseComposition :** Here the Underlying units are taken from the UnderlyingOpenCompostion and the US and EUR cash calculated according to the methodology

**Index Level File**

1. **fn\_StdComp2Level :** Generates P\_IndexLevel\_<VTID\_Date> using the Input File

C\_ CloseCompositionForLevel\_<VTID\_Date>.

Index Level is the sumproduct of the following fields :

1. Units
2. Price
3. FXRate
4. PriceFactor
5. FxFactor

### Business Logic – Index Calculation :

Rebalance – Every working day. Units are calculated every day

1. **The Index Level on each Index Business Day following the Index Start Date** :



1. **Determination of the Number of Units** :

Standard Unit Calculation : IndexLevel\*Weight/(Price\*FxRate)

1. **Determination of the Weight of Component Index :**

Weight = 1

|  |  |
| --- | --- |
| Indext | means the Index Value in respect of Index Valuation Date “t” |
| Indext-1 | means the Index Value in respect of the Index Valuation Date immediately preceding Index Valuation Date “t” |
| Pt | means the official closing level of the Constituent Index on Index Valuation Date “t”, as available on Bloomberg page MLFPMRET Index <HP> |
| Pt-1 | means Pt on Index Valuation Date “t-1” |
| FXt | means the Exchange Rate on Index Valuation Date “t” |
| FXt-1 | means FXt on Index Valuation Date “t-1” |
| Ratet-1 | means the Interest Rate as of the immediately preceding Index Valuation Date “t” |
| ACTt | means the number of calendar days from (but excluding) the immediately preceding Index Valuation Date to (and including) Index Valuation Date “t” |

1. **Cash Units :**
2. **Cash Open Units** :

Index Level(t) – Sumproduct(Constituent Open Units(t),Settlement Prices(t))

1. **Cash Close Units** :

Index Level(t) – Sumproduct(Constituent Close Units(t),Settlement Prices(t))

## Index – Introduction

**Name** : Nomura Interest Rate Swaps Momentum and Value Index

**Type** : Interest Rate

**Index VTId** : 1. Si:US:XX:XX:X:XX:SMV9:XXXX:NM

2. Si:US:XX:XX:X:XX:SMV3:XXXX:NM

3. Si:EU:XX:XX:X:XX:SMV9:XXXX:NM

4. Si:EU:XX:XX:X:XX:SMV3:XXXX:NM

**Index Sponsor** : Nomura

**Index Base Date** : 09 May 1990

**Index Base Currency** : U.S. Dollars for NMRSMV9U and NMRSMV3U. EUR for NMRSMV9E and NMRSMV3E

### Input Files

### Input Files for Composition Calculation:

1. C\_VTIndexSpecificData\_<VTId>\_<RunDate>
2. Required Fields
   1. PriceDate – Text convertible to date
   2. SpecificTicker – Text
   3. FundingRate – Float
   4. IndexLevel – Float
   5. FundingRate – Float
   6. Fee – Float
   7. FinalExposure - Float
   8. RebalancingTransactionCost - Float
3. C\_VTOpenComposition\_<VTId>\_<RunDate>
4. Required Fields
   1. SpecificTicker – Text
   2. Units – Float
   3. Date - Text convertible to date
5. C\_VTDailyPrices\_<VTId>\_<RunDate>
6. Required Fields
   1. PriceDate – Text convertible to date
   2. GenericTicker - Text
   3. Price - Float
7. C\_VTCloseComposition \_<VTId>\_<RunDate>
8. Required Fields
   1. SpecificTicker – Text
   2. Units – Float
   3. Date - Text convertible to date
9. C\_VTHolidayCalendar\_<VTId>\_<RunDate>
10. Required Fields
    1. SpecificTicker – Text
    2. Units – Float
    3. Date - Text convertible to date

**Input Files for Level Calculation:**

1. C\_CloseCompositionForLevel\_<VTId>\_<RunDate>
   * 1. Required Fields
        1. Units (Float)
        2. Price (Float)
        3. FXRate (Float)
        4. PriceFactor (Float)
        5. FxFactor (Float)
     2. SumProduct of the above mentioned Fields gives the Index Level which is published in P\_IndexLevel\_<VTID\_Date> File

### Output Files

#### Output Files for Composition Calculation:

1. P\_VTCloseComposition\_<VTId>\_<RunDate>
2. Required Fields
   * + 1. SpecificTicker – Text
       2. Units – Float
       3. Date - Text convertible to date
3. P\_VTOpenComposition\_<VTId>\_<RunDate>
4. Required Fields
   * + 1. SpecificTicker – Text
       2. Units – Float
       3. Date - Text convertible to date
5. P\_VTIndexSpecificData\_<VTId>\_<RunDate>
6. Required Fields
   * + 1. PriceDate – Text convertible to date
       2. Units – Float

**Output Files for Level Calculation:**

1. P\_IndexLevel\_<VTId>\_<RunDate>
2. Required Fields
3. Date (Text convertible to Date)
4. IndexVTId (Text)
5. IndexLevel (Float)

### Class – Python

**NomuraMomentumValueIndexCalculation**: This is the VT class under which the calculations for

NMRSMV9U, NMRSMV3U, NMRSMV9E, NMRSMV3E are coded

1. Required parameters for the Initialization of the class
2. **RunDate** : Date for which the Index level is calculated (from C# argument)
3. **LastRunDate** : Date previous to the Run Date (from C# argument)
4. **OpenDate** :The next trading date fetched from Daily Prices file
5. **DailyPrices** : Data collected from the input file C\_VTDailyPrices\_<VTId>\_<RunDate>
6. **IndexSpecificData** : Data collected from the input file C\_VTOpenComposition\_<VTId>\_<RunDate>
7. **CloseComposition :** Data collected from the input file C\_CloseComposition\_<VTId>\_<RunDate>
8. **OpenComposition :** Data collected from the input file C\_OpenComposition\_<VTId>\_<RunDate>
9. **NewCloseComposition :** Data collected from the input file C\_VTOpenComposition\_<VTId>\_<RunDate>
10. **NewOpenComposition :** Dataframe created for NewOpenComposition
11. **OpenIndexSpecificData :** Dataframe created for OpenIndexSpecificData
12. **HolidayCalendar :** Data collected from the input file C\_HolidayCalendar\_<VTId>\_<RunDate>
13. **IndexLevel :** Initiated from 0
14. **RebalFlag :** Initiated from 0
15. **IndexVolatilityScale :**  This is fetched from IndexSpecificData (VolatilityScaleLeverage)
16. **directory :** Directory where all the Input and Output files are stored
17. **IndexVTId :** VTId of the Index
18. **Log :** log module used to error handling
19. According to the Master Mapping there is one sub-process called for this **VtCompOpenClose** which calls two functions from the VT class, **fn\_VTCloseComp** and **fn\_VTOpenComp**
20. **fn\_VTCloseComp** : The output file **P\_VTCloseComposition\_<VTId>\_<RunDate>** is generated here. The following methods are called
    * + 1. **fn\_NomuraMVRebalCheck :** The check for Rebal date is done here based on the methodology given in the rulebook
        2. **fn\_NomuraMVCloseComposition :** The close composition and index level are calculated here. For the close composition a copy of OpenComposition is made except for cash units. The units for all other underlyings remain the same. After calculating the Index level for the RunDate. Then the close cash units are calculated using the Index Level for RunDate
21. **fn\_VTOpenComp** : The output files **P\_VTOpenComposition\_<VTId>\_<RunDate>** and **P\_VTIndexSpecificData\_<VTId>\_<RunDate>** are generated here. This function calls for two more methods
    * + 1. **fn\_NomuraMVOpenComposition:** The Open composition is calculated here. The Open Units are calculated according to the methodology given in Rule book which is different for Rebal and non – Rebal periods. Then Open Cash is calculated in both the cases using the Index Level of Run Date
        2. **fn\_NomuraMVOpenIndexSpecificData :** There is particular change in any interim calculations. Only the Index Level is updated for Run Date in the Index Specific Data

**Index Level File**

1. **fn\_StdComp2Level :** Generates P\_IndexLevel\_<VTID\_Date> using the Input File

C\_ CloseCompositionForLevel\_<VTID\_Date>.

Index Level is the sumproduct of the following fields :

1. Units
2. Price
3. FXRate
4. PriceFactor
5. FxFactor

### Business Logic – Index Calculation :

1. **The Index Level on each Index Business Day following the Index Start Date** :



1. **Raw Return :**



1. **Return :**



1. **ReturnMomentum :**



1. **ReturnValue :**



1. **Index Volatility Scale :**



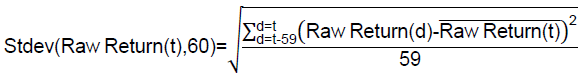
1. **Volatility factor :**



1. **Leverage(t) :**



1. **Stdev(Raw Return(t),60) :**



1. **Volatility Factor Cap :** The Volatility Factor Cap is computed as being the 75th percentile of Leverage(t) from and including 4th of January 2000 to and including 31st of December 2013
2. **Index Business Day** : Each day on which commercial banks and foreign exchange markets settle payments and are open for general business (including dealings in foreign exchange and foreign currency deposits) in London, New York and TARGET
3. **Cash Units :**
4. **Cash Open Units** :

Index Level(t) – Sumproduct(Constituent Open Units(t),Settlement Prices(t))

1. **Cash Close Units** :

Index Level(t) – Sumproduct(Constituent Close Units(t),Settlement Prices(t))

## Index – Introduction

**Name** : Nomura Interest Rate Swaps Momentum Index

**Type** : Interest Rate

**Index VTId** : 1. Si:US:XX:XX:X:XX:SMO3:XXXX:NM

2. Si:US:XX:XX:X:XX:SMO0:XXXX:NM

3. Si:US:XX:XX:X:XX:SMO9:XXXX:NM

**Index Sponsor** : Nomura

**Index Base Date** : 05 Feb 1990

**Index Base Currency** : U.S. Dollars

### Input Files

### Input Files for Composition Calculation:

1. C\_VTIndexSpecificData\_<VTId>\_<RunDate>
2. Required Fields
   1. PriceDate – Text convertible to date
   2. SpecificTicker – Text
   3. FundingRate – Float
   4. IndexLevel – Float
   5. FundingRate – Float
   6. Fee – Float
   7. FinalExposure - Float
   8. RebalancingTransactionCost - Float
3. C\_VTOpenComposition\_<VTId>\_<RunDate>
4. Required Fields
   1. SpecificTicker – Text
   2. Units – Float
   3. Date - Text convertible to date
5. C\_VTDailyPrices\_<VTId>\_<RunDate>
6. Required Fields
   1. PriceDate – Text convertible to date
   2. GenericTicker - Text
   3. Price - Float
7. C\_VTCloseComposition \_<VTId>\_<RunDate>
8. Required Fields
   1. SpecificTicker – Text
   2. Units – Float
   3. Date - Text convertible to date
9. C\_VTHolidayCalendar\_<VTId>\_<RunDate>
10. Required Fields
    1. SpecificTicker – Text
    2. Units – Float
    3. Date - Text convertible to date

**Input Files for Level Calculation:**

1. C\_CloseCompositionForLevel\_<VTId>\_<RunDate>
   * 1. Required Fields
        1. Units (Float)
        2. Price (Float)
        3. FXRate (Float)
        4. PriceFactor (Float)
        5. FxFactor (Float)
     2. SumProduct of the above mentioned Fields gives the Index Level which is published in P\_IndexLevel\_<VTID\_Date> File

### Output Files

#### Output Files for Composition Calculation:

1. P\_VTCloseComposition\_<VTId>\_<RunDate>
2. Required Fields
   * + 1. SpecificTicker – Text
       2. Units – Float
       3. Date - Text convertible to date
3. P\_VTOpenComposition\_<VTId>\_<RunDate>
4. Required Fields
   * + 1. SpecificTicker – Text
       2. Units – Float
       3. Date - Text convertible to date
5. P\_VTIndexSpecificData\_<VTId>\_<RunDate>
6. Required Fields
   * + 1. PriceDate – Text convertible to date
       2. Units – Float

**Output Files for Level Calculation:**

1. P\_IndexLevel\_<VTId>\_<RunDate>
2. Required Fields
3. Date (Text convertible to Date)
4. IndexVTId (Text)
5. IndexLevel (Float)

### Class – Python

**NomuraMomentumIndexCalculation**: This is the VT class under which the calculations for

NMRSMO3U, NMRSMO9U, NMRSMO0U are coded

1. Required parameters for the Initialization of the class
2. **RunDate** : Date for which the Index level is calculated (from C# argument)
3. **LastRunDate** : Date previous to the Run Date (from C# argument)
4. **OpenDate** :The next trading date fetched from Daily Prices file
5. **DailyPrices** : Data collected from the input file C\_VTDailyPrices\_<VTId>\_<RunDate>
6. **IndexSpecificData** : Data collected from the input file C\_VTOpenComposition\_<VTId>\_<RunDate>
7. **CloseComposition :** Data collected from the input file C\_CloseComposition\_<VTId>\_<RunDate>
8. **OpenComposition :** Data collected from the input file C\_OpenComposition\_<VTId>\_<RunDate>
9. **NewCloseComposition :** Data collected from the input file C\_VTOpenComposition\_<VTId>\_<RunDate>
10. **NewOpenComposition :** Dataframe created for NewOpenComposition
11. **OpenIndexSpecificData :** Dataframe created for OpenIndexSpecificData
12. **HolidayCalendar :** Data collected from the input file C\_HolidayCalendar\_<VTId>\_<RunDate>
13. **IndexLevel :** Initiated from 0
14. **RebalFlag :** Initiated from 0
15. **IndexVolatilityScale :**  This is fetched from IndexSpecificData (VolatilityScaleLeverage)
16. **directory :** Directory where all the Input and Output files are stored
17. **IndexVTId :** VTId of the Index
18. **Log :** log module used to error handling
19. According to the Master Mapping there is one sub-process called for this **VtCompOpenClose** which calls two functions from the VT class, **fn\_VTCloseComp** and **fn\_VTOpenComp**
20. **fn\_VTCloseComp** : The output file **P\_VTCloseComposition\_<VTId>\_<RunDate>** is generated here. The following methods are called
    * + 1. **fn\_NomuraMRebalCheck:** The check for Rebal date is done here based on the methodology given in the rulebook
        2. **fn\_NomuraMCloseComposition:** The close composition is calculated here. For the close composition a copy of OpenComposition is made except for cash units. The units for all other underlyings remain the same.
        3. **fn\_NomuraMCalculateIndexLevel :** The Index Level for the Run Date is calculated using the methodology given in the rule book. This method takes data from Daily Prices, IndexSpecific Data and Holiday Calender
        4. **fn\_NomuraMCalculateCash :** As the Index Level for the Run date is calculated in the above function. Here the Close Cash is calculated ucing the Index Level for Run Date. This takes input from NewClosecomposition data frame
21. **fn\_VTOpenComp** : The output files **P\_VTOpenComposition\_<VTId>\_<RunDate>** and **P\_VTIndexSpecificData\_<VTId>\_<RunDate>** are generated here. This function calls for two more methods
    * + 1. **fn\_NomuraMOpenIndexSpecificData:** The Open Index Specific data is calculated here i.e., all the interim calculations like calculation of signal, returns, gross index levels. According to the Rebal flag and methodology in the rule book the interim data is calculated here
        2. **fn\_NomuraMOpenComposition:** The Open Composition is calculated here. Open Units are calculated according to the methodology given in rule book. This function takes input from above function for Interim data

**Index Level File**

1. **fn\_StdComp2Level :** Generates P\_IndexLevel\_<VTID\_Date> using the Input File

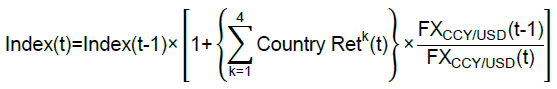
C\_ CloseCompositionForLevel\_<VTID\_Date>.

Index Level is the sumproduct of the following fields :

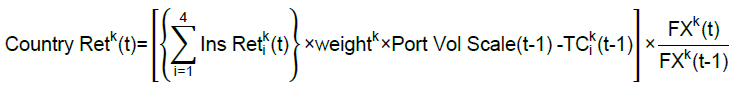
1. Units
2. Price
3. FXRate
4. PriceFactor
5. FxFactor

### Business Logic – Index Calculation :

1. **The Index Level on each Index Business Day following the Index Start Date** :



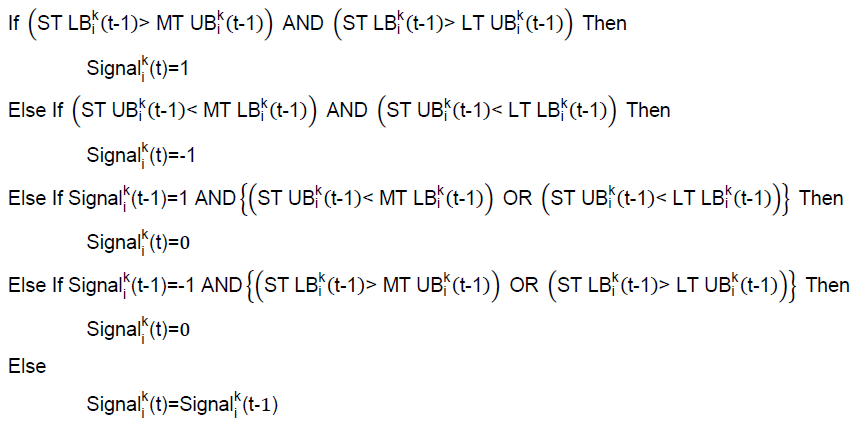
1. **Country Retk:**



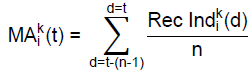
1. **Ins Retk :**



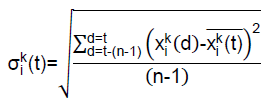
1. **Signalik :**



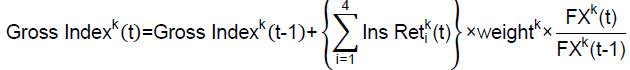
1. **MA :**



1. **Sigma:**



1. **Gross Indexk :**



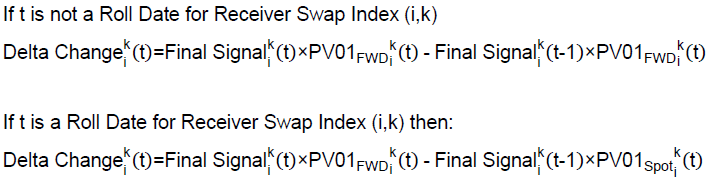
1. **Leverage(t) :**



1. **Outright TCk:**



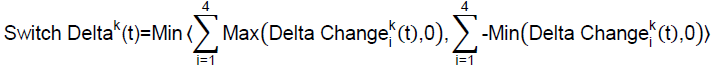
1. **Delta Changeik :**



1. **Final Signalik** :



1. **Switch TC :**



1. **Roll TC:**



1. **Cash Units :**
2. **Cash Open Units** :

Index Level(t) – Sumproduct(Constituent Open Units(t),Settlement Prices(t))

1. **Cash Close Units** :

Index Level(t) – Sumproduct(Constituent Close Units(t),Settlement Prices(t))

## Bank Calculated Indices

### Following are the Bank Calculated Indices

|  |  |
| --- | --- |
| Ei:EU:DT:XX:X:XX:SEDE:XXXX:DB | DBCUSEDE |
| Ei:US:DT:XX:X:XX:SEDU:XXXX:DB | DBCUSEDU |
| Fi:US:DT:XX:X:XX:STGU:XXXX:BP | BPFXSTGU |
| Ei:EU:DT:XX:X:XX:MRET:XXXX:ML | MLFPMRET |
| Ei:EU:DT:XX:X:XX:SDIV:XXXX:ML | MLEISDIV |
| Ei:EU:DT:XX:X:XX:DIVI:XXXX:ML | MLEIDIVI |
| Ei:EU:DT:XX:X:XX:LSD2:XXXX:ML | MLEILSD2 |
| Ei:US:DT:XX:X:XX:MRUE:XXXX:ML | MLFPMRUE |
| Ci:US:XX:XX:X:XX:NBXA:XXXX:BP | NBXAER |
| Vi:US:XX:XX:X:XX:IXER:XXXX:SG | DSVIXER |
| Ci:US:XX:XX:X:XX:LNEN:XXXX:CI | CVICLNEN |
| Ri:US:XX:XX:X:XX:EPUG:XXXX:BR | BXIIEPUG |
| Ri:US:XX:XX:X:XX:EP3G:XXXX:BR | BXIIEP3G |
| Si:US:XX:XX:X:XX:PRGU:XXXX:BR | BXIIPRGU |
| Si:US:XX:XX:X:XX:TSTP:XXXX:BR | BXIITSTP |
| Ei:US:XX:XX:X:XX:DWSP:XXXX:ML | MLEIDWSP |
| Vi:US:XX:XX:X:XX:RDUE:XXXX:CS | CSVPRDUE |

### Classes defined for the Composition and Level calculation

Input Files for Composition calculation:

1. C\_BankOCRawData\_<VTID>\_<RunDate>
   * + 1. **Required Fields**
          1. IndexVTId - Text
          2. InstrumentVTId - Text
          3. Date – Text convertible to date
          4. CurrencyId - Number
          5. CurrencyVTId - Text
          6. Currency - Text
          7. BankUnit - Float
          8. BankWeight - Float
          9. BankPrice - Float
          10. BankFxRate - Float
          11. BankIndexLevel - Float
          12. OpenDate – Text convertible to Date
          13. IndexLevel - Float
          14. Price - Float
          15. PriceSourceId - Integer
          16. PriceSource - Text
          17. FxRate - Float
          18. FxName - Text
          19. FxSourceId - Number
          20. FxSource - Text
          21. LastRunDate – Text convertible to Date
          22. LastRunPrice - Float
2. C\_CloseComposition\_<VTID>\_<RunDate>
   * + 1. **Required Fields**
          1. IndexVTId - Text
          2. InstrumentVTId – Text
          3. SpecificTicker - Text
          4. GenericTicker - Text
          5. SecurityId - Number
          6. MaturityDate – Text Convertible to Date
          7. Date – Text convertible to Date
          8. CurrencyId - Number
          9. Currency - Text
          10. Units - Float
          11. Price - Float
          12. PriceModel – Text
          13. PriceFactor - Number
          14. PriceSource – Text
          15. PriceSourceId - Number
          16. FxName - Text
          17. FxRate - Float
          18. FxModel – Text
          19. FxFactor - Number
          20. FxSource – Text
          21. FxSourceId - Number
          22. Source – Text
3. C\_OpenComposition\_<VTID>\_<RunDate>
   * + 1. **Required Fileds**
          1. IndexVTId - Text
          2. InstrumentVTId – Text
          3. SpecificTicker - Text
          4. GenericTicker - Text
          5. SecurityId - Number
          6. MaturityDate – Text Convertible to Date
          7. Date – Text convertible to Date
          8. CurrencyId - Number
          9. Currency - Text
          10. Units - Float
          11. Price - Float
          12. PriceModel – Text
          13. PriceFactor - Number
          14. PriceSource – Text
          15. PriceSourceId - Number
          16. FxName - Text
          17. FxRate - Float
          18. FxModel – Text
          19. FxFactor - Number
          20. FxSource – Text
          21. FxSourceId - Number
          22. Source – Text

Output Files for Composition calculation:

1. P\_CloseComposition\_<VTID>\_<RunDate>
   * + 1. **Required Fields:**
          1. IndexVTId - Text
          2. InstrumentVTId – Text
          3. SpecificTicker - Text
          4. GenericTicker - Text
          5. SecurityId - Number
          6. MaturityDate – Text Convertible to Date
          7. Date – Text convertible to Date
          8. CurrencyId - Number
          9. Currency - Text
          10. Units - Float
          11. Price - Float
          12. PriceModel – Text
          13. PriceFactor - Number
          14. PriceSource – Text
          15. PriceSourceId - Number
          16. FxName - Text
          17. FxRate - Float
          18. FxModel – Text
          19. FxFactor - Number
          20. FxSource – Text
          21. FxSourceId - Number
          22. Source – Text
2. P\_OpenComposition\_<VTID>\_<RunDate>
   * + 1. **Required Fileds**
          1. IndexVTId - Text
          2. InstrumentVTId – Text
          3. SpecificTicker - Text
          4. GenericTicker - Text
          5. SecurityId - Number
          6. MaturityDate – Text Convertible to Date
          7. Date – Text convertible to Date
          8. CurrencyId - Number
          9. Currency - Text
          10. Units - Float
          11. Price - Float
          12. PriceModel – Text
          13. PriceFactor - Number
          14. PriceSource – Text
          15. PriceSourceId - Number
          16. FxName - Text
          17. FxRate - Float
          18. FxModel – Text
          19. FxFactor - Number
          20. FxSource – Text
          21. FxSourceId - Number
          22. Source – Text

Input Files for Index Level calculation:

1. C\_ CloseCompositionForLevel\_<VTID>\_<RunDate>
   * + 1. Required Fields
          1. Units (Float)
          2. Price (Float)
          3. FXRate (Float)
          4. PriceFactor (Float)
          5. FxFactor (Float)

Output Files for Index Level calculation:

1. P\_ IndexLevel\_<VTID>\_<RunDate>
   * + 1. Required Fields
          1. Date (Text convertible to Date)
          2. IndexVTId (Text)
          3. IndexLevel (Float)

For all the bank calculated Indices the calculation is done under three classes

1. **cl\_BankOCClose** : This class contains calculation for Close Composition for all the Bank Calculated Indices except for Option Indices. This class contains the following functions
   * 1. **fn\_Stdclose** : This function is used in calculation of close composition for the following Five Indices
        1. Fi:US:DT:XX:X:XX:STGU:XXXX:BP
        2. Ei:EU:DT:XX:X:XX:MRET:XXXX:ML
        3. Ei:EU:DT:XX:X:XX:SDIV:XXXX:ML
        4. Ei:EU:DT:XX:X:XX:DIVI:XXXX:ML
        5. Ei:EU:DT:XX:X:XX:LSD2:XXXX:ML

This function takes data from OpenComposition file, makes a copy of it except for the Cash Units. Then the Close Composition is created. Close Cash Units are also calculated here

* + 1. **fn\_DBClose** : This function is used in calculation of close composition for the following two indices
       1. Ei:EU:DT:XX:X:XX:SEDE:XXXX:DB
       2. Ei:US:DT:XX:X:XX:SEDU:XXXX:DB

This function is takes data from BankOCRawData and CloseComposition files. The units for all the Instruments are taken from BankOCRawData and the cash units are calculated using the BankOCRawData

* + 1. **fn\_MRUEClose** : This function is used in calculation of close composition for the following index
       1. Ei:US:DT:XX:X:XX:MRUE:XXXX:ML

This function takes data from OpenComposition and BankOCRawData. Here both USD Close Cash and EUR Close cash are calculated. The Units of the underlying instruments are taken from OpenComposition

* + 1. **fn\_CloseBankOCwoCash** : This function is used in calculation of close composition for the following Five Indices
       1. Ci:US:XX:XX:X:XX:NBXA:XXXX:BP
       2. Ri:US:XX:XX:X:XX:EPUG:XXXX:BR
       3. Ri:US:XX:XX:X:XX:EP3G:XXXX:BR
       4. Si:US:XX:XX:X:XX:PRGU:XXXX:BR
       5. Si:US:XX:XX:X:XX:TSTP:XXXX:BR

This function takes data from OpenComposition and BankOCRawData. The Bank data does not contain any cash in their

* + 1. **fn\_IXERClose:** This function is used in calculation of close composition for the following index
       1. Vi:US:XX:XX:X:XX:IXER:XXXX:SG

This function takes data from BankOCRawData and calculates the units of all the underlying Instruments. Close Cash Units are also calculated here.

* + 1. **fn\_LNENClose:** This function is used in calculation of close composition for the following index
       1. Ci:US:XX:XX:X:XX:LNEN:XXXX:CI

This function takes data from BankOCRawData and calculates the units for all the Underlying Instruments. Close Cash Units are also calculated here

* + 1. **fn\_RDUEClose:** This function is used in calculation of close composition for the following index
       1. Vi:US:XX:XX:X:XX:RDUE:XXXX:CS

This function takes data from BankOCRawData and calculates the units for all the Underlying Instruments. Close Cash Units are also calculated here

1. **cl\_BankOCOpen** : This class contains calculation for Open Composition for all the Bank Calculated Indices except for Option Indices
   * 1. **fn\_StdOpen:** This function is used in calculation of close composition for the following Five Indices
        1. Fi:US:DT:XX:X:XX:STGU:XXXX:BP
        2. Ei:EU:DT:XX:X:XX:MRET:XXXX:ML
        3. Ei:EU:DT:XX:X:XX:SDIV:XXXX:ML
        4. Ei:EU:DT:XX:X:XX:DIVI:XXXX:ML

This function takes data from BankOCRawData and OpenComposition file

* + 1. **fn\_IoIOpenSameCurrency:** This function is used in calculation of close composition for the following Five Indices
       1. Ei:EU:DT:XX:X:XX:LSD2:XXXX:ML

This function takes data from BankOCRawData and OpenComposition file. Open Units for Instruments and Cash are calculated

* + 1. **fn\_IoIMRUEOpen:** This function is used in calculation of close composition for the following Five Indices
       1. Ei:US:DT:XX:X:XX:MRUE:XXXX:ML

This function takes data from BankOCRawData and OpenComposition file. Open Units for Instruments and Cash are calculated

* + 1. **fn\_OpenBankOCwoCash:** This function is used in calculation of close composition for the following Five Indices
       1. Ci:US:XX:XX:X:XX:NBXA:XXXX:BP

This function takes data from BankOCRawData and OpenComposition file. Open Units for Instruments and Cash are calculated

* + 1. **fn\_BarclaysExceedOpen:** This function is used in calculation of close composition for the following Five Indices
       1. Ri:US:XX:XX:X:XX:EPUG:XXXX:BR
       2. Ri:US:XX:XX:X:XX:EP3G:XXXX:BR

This function takes data from BankOCRawData and OpenComposition file. Open Units for Instruments and Cash are calculated for Barclays Indices

* + 1. **fn\_BarclaysExcessReturnOpen:** This function is used in calculation of close composition for the following Five Indices
       1. Si:US:XX:XX:X:XX:PRGU:XXXX:BR
       2. Si:US:XX:XX:X:XX:TSTP:XXXX:BR

This function takes data from BankOCRawData and OpenComposition file. Open Units for Instruments and Cash are calculated for Barclays Excess Return Indices. Methodology for calculation of units is different

* + 1. **fn\_RDUEOpen:** This function is used in calculation of close composition for the following index
       1. Vi:US:XX:XX:X:XX:RDUE:XXXX:CS

This function takes data from BankOCRawData and Close Compostion , calculates the units for all the Underlying Instruments. Close Cash Units are also calculated here for RDUE Index

1. **cl\_Comp2Level** : This class contains calculation for Final Index calculation for all the Indices
   * 1. **fn\_StdComp2Level:** The Final Index Level calculation for all the Bank and VT indices is done here except for Equity Indices and Option Indices. It is the sumproduct of Prices, Units and FX Rates that are fetched from the CloseCompositionForLevel file
     2. **fn\_EqComp2Level:** The Final Index calculation for the Equity Indices is done here. This function takes input from two files, CloseCompositionForLevel and IndexCalculationDetailsForLevel.

### Classes defined for the Composition and Level calculation for Option Indices

Indices Listed under this class:

1. Ei:US:XX:XX:X:XX:DWSP:XXXX:ML
2. **cl\_OptionMetricsCalculations** : The Bank calculation (in this case price calculation for options), both close and open composition for Option Indices are done here. All the Option Metrics calculations such as Volatility Surface, Forward Curve, Option Pricing are done here. OptionMetrics module is imported.
   * + 1. **fn\_VolSurfaceCalc :** This function imports methods from OptionMetrics Module for Implied Volatility surface calculation for these particular Indices
       2. **fn\_FwdCurveCalc:** This function imports methods from OptionMetrics Module for Forward Price calculation for these particular Indices
       3. **fn\_OptionPricingCalc:** This function imports methods from OptionMetrics Module for Implied Volatility surface calculation for these particular Indices
       4. **fn\_EquityOptionBankOpen:** This function is used for the calculation of Open Composition for the Equity Option Index. This function provides the facility to take data directly from the BankOCRawData file or else if some data are missing in this file, they can be fetched from other input files. Both these are handled in the CoreOptionPricing module. All the calculations such as greeks, forward price, implied volatility for open composition are calculated here. This function generates the C\_OpenComposition file the Equity Option Indices
       5. **fn\_PnLCalculator :** This function basically calculates the Impact of change in Security Price, Interest Rate, Volatility, Time and also the Unexplained PnL. This takes data for two days and calculates the option price by changing any one of the above parameters and then interprets the change in option price. This gives the PnL calculations
       6. **fn\_EquityOptionBankClose :** This function is used for the calculation of Close Composition for the Equity Option Index. This function provides the facility to take data directly from the Open Composition file or else if some data are missing in this file, they can be fetched from other input files. Both these are handled in the CoreOptionPricing module. All the calculations such as greeks, forward price, implied volatility for open composition are calculated here. This function generates the C\_CloseComposition file the Equity Option Indices
3. **cl\_Comp2Level** : This class contains calculation for Final Index calculation for all the Option Indices also
   * + 1. **fn\_IndexRisk :** This is the final function where the Risk parameters for the corresponding Equity Option Index are fetched and sent to the file P\_IndexRiskParameters
       2. **fn\_OptionComp2Level :** The Final Index calculation for the Option Indices is done here. This function takes input from CloseCompositionForLevel and the Index Level is calculated using StdComp2Level. Both the outputs from IndexRisk and StdComp2Level are put together here.