Project ID:

Native RFL Factors in Quartz

Apr 2, 2012

A Major Qualifying Project submitted to the faculty of
Worcester Polytechnic Institute in partial fulfillment of the requirements for the Bachelor of Science degree

Chenchen Zhang

_____

Luyang Zhang

_____

Advisors

Professor Jon Abraham
Department of Mathematical Sciences

Professor Micha Hofri
Department of Computer Science

Table of Contents

Abstract

The Global Markets Research Technology Department at Bank of America is on track to launch the next generation, cross-asset technology platform Quartz. Previously, each line of business at Bank of America operated on its own technology platform, which caused duplication of large distributive databases. With the new Quartz platform, Bank of America will alleviate the cost of replication by introducing Sandra database, a globally replicated object store. In this project, our team focused on the database migration of CDO (Collateralized Debt Obligation) square Calibration tool, an in-house proprietary software tool used to price and analyze this certain credit derivative. We defined the new data structure of CDO square, which is fully compatible with Quartz, and provided web services so that CDO square objects are accessible to other applications on the Quartz platform. We proposed and implemented the well-tested framework within which square Calibration tool can communicate with the new Sandra database with comparably low overhead and high stability.

Acknowledgements

Our group would likely to acknowledge many people for the chance to take part in the project and the accomplishments result from it.

First, we are deeply grateful for our professors from Worcester Polytechnic Institute for all of their input and support - Professor Jon Abraham, and Professor Micha Hofri.

We also wanted to acknowledge our sponsor in Bank of America, Ron Toam. Lastly we would like to thank our supervisor during the project work, Hyunmin Lee. We truly appreciate the many hours of guidance and support she provided us during our time at Bank of America.

Authorship Page

Chenchen Zhang and Luyang Zhang contributed equally writing the report, completing the methodology, and developing our final product.

Table of Figures

List of Tables

Executive Summary

Bank of America's Structured Credit Technology (SCT) group is a global development and support team located in New York City and London, which provides effective and efficient end-to-end technology support to the Structured Credit Trading Desk. The team is responsible for the development and maintenance of several proprietary applications used by traders in Bank of America.

The main goal of the WPI project team is to improve and upgrade BoA's CDO^2 Calibration Tool to follow the directive that in the future all the bank's applications should reside on the Quartz platform. It can be divided into two objectives.

The first objective is to migrate database support from Camden to Sandra. Before Quartz was introduced into Bank of America several years ago, Camden was the only database used to store all CDO^2 parameters and calculation results of the application. It is becoming a legacy database, as we gradually move all data to Sandra, an objective database that is more compatible with Quartz.

Then we need to address the issue of how Risk Engine, where all calculation results are produced, can have access to the data in Sandra. Our goal is to design a convenient and efficient method without causing too much overhead on the database-side. The solution we provided and implemented is setting up a REST service API for Risk Engine that will return the JSON format of requested objects. It enjoys an advantage over others by only using the basic HTTP methods and avoiding permission issues to have access to Sandra.

For the current time being, it is still of Bank of America's best interest to publish CDO^2 calculation results to both Camden and Sandra databases for stability issue. After

discussing with the development team and analyzing all possible situations we may encounter during the publishing process, we implemented specific error handling cases to prevent inconsistency in database in the case of any database failure.

Chapter One: Introduction

After the merger with Merrill Lynch, Bank of America has been addressed the issue of efficient communication among offices all over the world. The typical tactical approach to achieve integration via merger, silo, and transition no longer fits in our time of uncertain markets, legislative demands, and regulatory scrutiny, and calls for a strategic approach that defines cross firm standard based on common reference data, enhanced client offerings, and an unified, cross-platform risk/trading platform. In the first year of strategic vision progress, Bank of America focuses on putting together teams, design architecture, coding and prototyping, and documentation. Currently, Bank of America enters into the second stage of strategic vision that stresses adoption, pilots and education.

Quartz platform is introduced with all trades, market data, analytics, and risk measures that help Bank of America to improve pricing, risk management and use of capital. Quartz is designed with quick turnaround for maintainable instruments, pricing, risk management, lifecycle support, settlements, and approval workflow.

As part of the structure credit trading department, our project group is responsible for contributing to the credit trading application within Quartz platform by implementing and modifying functionalities of existing applications. The whole development process strictly follows the general standard and convention of Quartz development process, which includes implementing, testing, checking into repository, reviewing and finally pushing into production environment.

Chapter Two: Background

2.1 Bank of America and Global Markets Research Technology

Bank of America is one of the world's largest financial institutions, serving individual consumers, small- and middle-market businesses and large corporations with a full range of banking, investing, asset management and other financial and risk management products and services.

The company provides unmatched convenience in the United States, serving approximately 56 million consumer and small business relationships with approximately 5,600 retail banking offices and approximately 16,200 ATMs and award-winning online banking with 30 million active users. (Bank of America Overview, 2012)

Bank of America is among the world's leading wealth management companies and is a global leader in corporate and investment banking and trading across a broad range of asset classes, serving corporations, governments, institutions and individuals around the world. Bank of America offers industry-leading support to approximately 4 million small business owners through a suite of innovative, easy-to-use online products and services. The company serves clients through operations in more than 40 countries. Bank of America Corporation stock (NYSE: BAC) is a component of the Dow Jones Industrial Average and is listed on the New York Stock Exchange. (Bank of America Overview, 2012)

Global Markets and Research Technology & Operations (GMRT&O) is a division under Global Technology & Operations in Bank of America. It provides end-to-end technology solutions and operations support for the Global Markets businesses including Equity, Electronic Trading, Rates & Currencies, Credit & Structured Products,

Commodities, Research, Sales and Capital Markets. In addition, the group is responsible for establishing an Architecture and Strategy framework for consistency across the Global Markets platforms. (Global Markets and Risk Technology (GMRT), 2012)

2.2 Quartz

2.2.1 Quartz Platform

Quartz is the next generation, cross-asset technology platform for Bank of America Merrill Lynch Global Markets.

As technology becomes the key in times of uncertain markets, legislative demands, and regulatory scrutiny, Bank of America Merrill Lynch Global Markets attempts to improve pricing, risk management, and use of capital along with the traditional algorithm trading. (Quartz Academy - Overview Session, 2010)

Quartz platform integrates trades, market data, analytics, and risk measure functionalities across all asset groups. Several of the major Quartz components that we have used in the implementation process include: (Relevant Quartz Components, 2010)

- QzDesktop: the launchpad for globally distributed applications
- QzDev: the Quartz IDE to develop Python code in
- Sandra: the Quartz object-based data store
- HUGS: the Quartz grid scheduler to run code in parallel
- Bob: the Quartz scheduling agent to run jobs
- QzTable: models large datasets from various sources
- AMPS: a high performance messaging system, utilized by Quartz

The Quartz Desktop will bootstrap the quartz platform on your windows machine. It ensures that you are always running the latest version of the quartz platform. Each application is represented as an icon within a folder in quartz desktop. An icon simply points to a python script in the source database which implements your application. (QzDesktop , 2010)

QZDev is Quartz's integrated development environment (IDE). The key integrated features include ability to write python code using the Quartz core libraries as well as standard Python libraries, an integrated Python Shell with a built-in debugger, a central source control repository, an ability to search and reuse all source to all Quartz applications and tools, and an integrated agile process with a code-review process.

Sandra is a multithreaded C++ server which runs primarily on Linux. The Quartz development team writes its own database server to support features like: (Sandra Features, 2010)

- Support for thousands of clients (for grid computing)
- Minimizing read/write contention by using optimistic locking and foregoing read isolation.
- Seamless schema migration. Support for lazy migration of old schemas.
- Globally synchronized via log replication.
- Transaction log exposed as a first class api for a consistent way to support notifications/auditing/replication.
- Multiple write masters across WAN links, no single point of failure
- Optimistic locking

- Conflict resolution at an object/transaction granularity
- Authentication/entitlements as yourself instead of generic db admin accounts

2.2.2 Quartz Development Process

The three primary goals for the Quartz development process are, a minimum learning curve for application development, provide ultra-fast turnaround, and ship robust, high-quality applications.

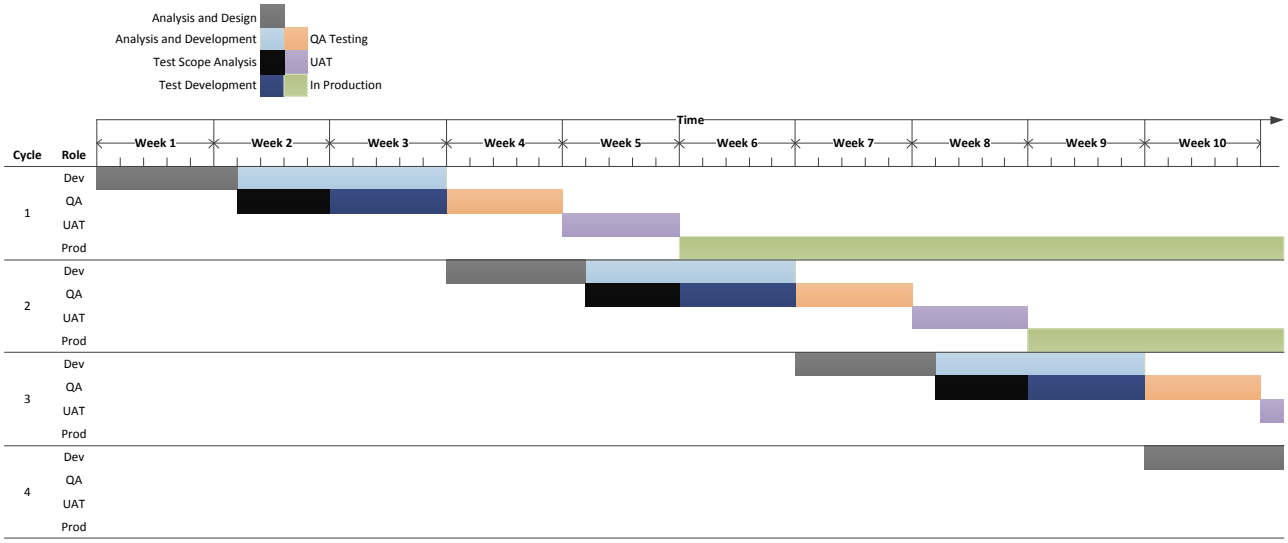


Figure 1: Overview of Release Cycle

The development cycle contains seven phases:

- The Analysis and Design phase, where the developer identifies the necessary functionality and integrate them into the system
- The Test Scope Analysis, where QA analyze the range of tests needed for the new functionality.

- Development phase, where the developer implements the needed functionality.
- The Test Development phase, where QA defines tests and implements automated tests to achieve the needed testing scope.
- The QA Testing phase, where the testing is made and bugs are identified and escalated.
- The UAT phase, where the User gets involved and signs off the added functionality.
- The Production phase, where the functionality has entered the live production environment.

2.2.3 Quartz Coding Styles

The coding standards employed by the Quartz team are based on the recommendations within the PEP 8 - Style Guide.

The Quartz project follows the general Python coding style. A few relevant rules from the guide:

- Use 4 space tabs per indentation level. Never mix tabs and spaces
- Imports should be on separate lines
- Avoid spaces immediately inside parentheses, brackets or braces
- Avoid naked exceptions
- 'Single-quote strings when possible'

2.2.4 Version Control

All production code is stored in the Sandra database referred to source, which contains local replicas in various geographical areas. Although every file edited by developer is loaded from source, when developer finishes the implementation and tries to save the changes, the modified file will be saved in developer's own user area, which lives in a database called "homedirs".

Any time QzDev runs Python code and needs to load a new module, it will look inside "homedirs" first, and then in source. It will do that for all users, and each user has their own user area in "homedirs".

2.3 Structured Credit Technology

By definition, structured credit trading usually refers to products consists of different tranches of portfolios of credit instruments. Common types of structured credit products include cash CDOs, synthetic CDOs, and nth-to-default baskets. (Structured credit, 2012)

In order to price CDO, Bank of America is currently developing a tool named "CDO2 Calibration Tool" to retrieve the raw data from Sandra database and send the data into risk engine, which calculates the results based on the random factor loading model.

Table 1 below describes different sub-applications existing within the Quartz that that are developed by the Structured Credit Technology team at Bank of America.

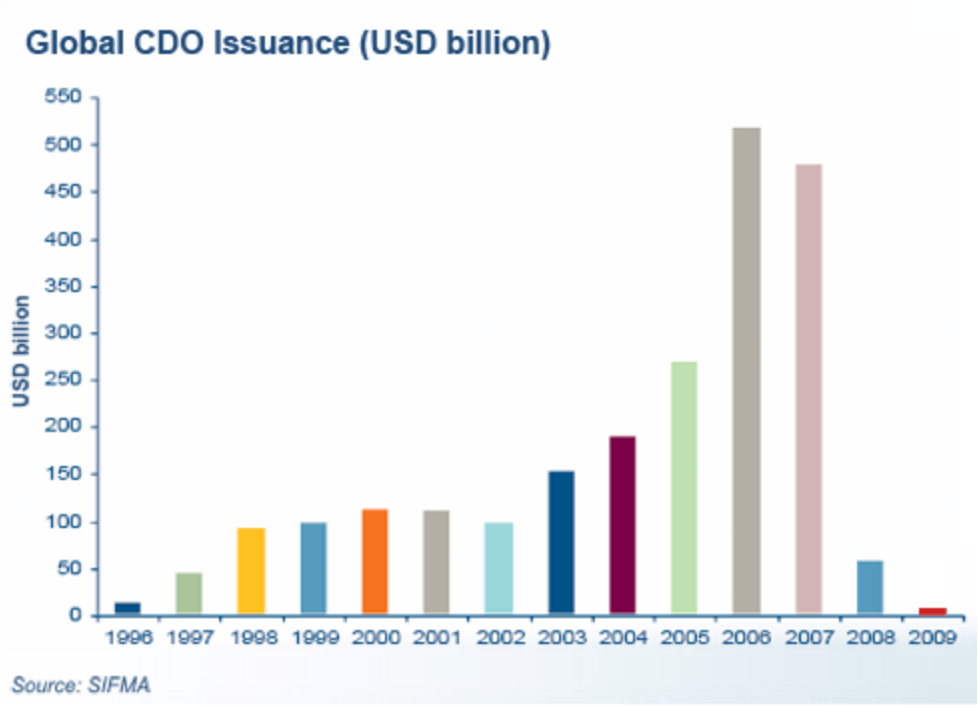
Application	Description
Cash and CDS Volume By Date	The application will support similar functionality as excel pivot tables. The user will be able specify what columns will be

	grouped by across both the vertical and horizontal dimension.
Index Options Market Making	A phased approach for adopting Index Options into Quartz. Phase 1: will involve migrating ivol functionality to Quartz and updating the market data model used to store data Phase 2: Build out a Index Option pricing/marking tool which leverages pricing and calibration of upfronts-> vols Phase 3: feed external systems
Bond Option Vol Upload Tool	The Bond Option Vols Publish application provides users with functionality to interface with Camden. Users can retrieve vol levels, edit it and publish edited data to Camden.
CDO2 Calibration/Marking Tool	Interfaces directly with Risk Engine and calculates RFL calibration parameters for CDO^2 trades and these results can be written into Sandra database.

Table 1: Structured Credit Technology Internal Application

2.4 CDO and CDO^2

CDO, which stands for collateralized debt obligation, is a type of structured asset-based security that is issued by special entities and collateralized by debt obligations, most of the time high-risk and high-yield bonds and loans. CDOs were seen to be flourishing during 2000 – 2007 and became an extremely high-profit credit derivative for investment banks. But then they suffered great losses and have been almost destroyed in the subprime mortgage crisis, because of the unabatedly growing issuance of CDOs and the declining quality of their collateral of which a large proportion is subprime bonds. It is estimated that CDOs take responsibility for nearly 542 billion dollars in write-downs for investment banks since the start of financial crisis. (Katherine, 2009)



In spite of the fact that CDO collapsed and investors flee from this area during subprime crisis, CDO is still of interest to the market. In 2010, Citigroup became the first investment bank underwriter for. It is also reported that JP Morgan, Bank of America and Deutsche Bank are approaching managers of leveraged loans to offer terms for new CDOs. (Bloomberg, 2010)

CDO-squared is identical to a CDO except for the asset securing the obligation. (CDO-Squared, 2012) Instead of backing by a pool of bonds, loans and other credit instruments, CDO-squared are backed by other CDO tranches. Namely, A CDO-squared is a CDO of a CDO. The underlying collateral consists of single tranches of CDOs or a mixed pool of CDO tranches and asset–backed securities. Banks could resell the credit risk they get in CDOs by issuing CDO-squared. The first CDO-squared deal was the USD 343m Zais Investment Grade (Zing I) deal in 1999.

2.5 CDO Pricing

2.5.1 Gaussian Copula

In probability theory and statistics, a copula is a kind of distribution function that is commonly used to describe the dependence between random variables. Copula gains its popularity by allowing easy modeling and estimation the distribution of random vectors by separating the estimation of marginal and copula. (Copula (probability theory), 2012)

Consider a random vector $(X_1, X_2, \ldots, X_d)$. Suppose its margins are continuous, i.e. the marginal CDFs $F_i(x) = \mathbb{P}[X_i \leq x]$ are continuous functions. By applying the probability integral transform to each component, the random vector $(U_1, U_2, \ldots, U_d) = (F_1(X_1), F_2(X_2), \ldots, F_d(X_d))$ has uniform margins. The copula of $(X_1, X_2, \ldots, X_d)$ is defined as the joint cumulative distribution function of $(U_1, U_2, \ldots, U_d)$: (Copula (probability theory), 2012)

$$C(u_1, u_2, \ldots, u_d) = \mathbb{P}[U_1 \leq u_1, U_2 \leq u_2, \ldots, U_d \leq u_d].$$

Gaussian copula is a distribution over the unit cube $[0, 1]^d$. Suppose we have a correlation matrix $\Sigma \in \mathbb{R}^{d \times d}$, the Gaussian copula with parameter matrix Σ can be expressed as

$$C_\Sigma^{Gauss}(u) = \Phi_\Sigma \left(\Phi^{-1}(u_1), \ldots, \Phi^{-1}(u_d) \right),$$

where Φ^{-1} represents the inverse cumulative distribution function of a standard normal and is the joint cumulative distribution function of a multivariate normal distribution with mean vector zero and covariance matrix identical to the correlation matrix.

The density of Gaussian copula can be expressed as (Arbenz, 2011)

$$c_{\Sigma}^{Gauss}(u) = \frac{1}{\sqrt{\det \Sigma}} \exp\left(-\frac{1}{2}\begin{pmatrix}\Phi^{-1}(u_1)\\ \vdots \\ \Phi^{-1}(u_d)\end{pmatrix}^{T} \cdot \left(\Sigma^{-1} - \mathbf{I}\right) \cdot \begin{pmatrix}\Phi^{-1}(u_1)\\ \vdots \\ \Phi^{-1}(u_d)\end{pmatrix}\right),$$

where $\mathbf{I}$ is the identity matrix.

To help understand the concept of Gaussian Copula, it would be helpful to illustrate with a two-dimension Gaussian Copula. (Schmidt, 2006)

$$C_{\rho}^{Ga}(u_1, u_2) = \Phi_{\Sigma}(\Phi^{-1}(u_1), \Phi^{-1}(u_2)),$$

where Σ is the 2 × 2 matrix with 1 on the diagonal and ρ otherwise. Φ denotes the cdf of a standard normal distribution while Φ_{Σ} is the cdf for a bivariate normal distribution with zero mean and covariance matrix Σ. Note that this representation is equivalent to

$$\int_{-\infty}^{\Phi^{-1}(u_1)} \int_{-\infty}^{\Phi^{-1}(u_2)} \frac{1}{2\pi\sqrt{1-\rho^2}} exp(-\frac{s_1^2 - 2\rho s_1 s_2 + s_2^2}{2(1-\rho^2)}) ds_1 s_2$$

2.5.2 CDO Pricing using Gaussian Copula

In the paper "On Default Correlation: A Copula Function Approach", Gaussian copula is firstly applied to CDOs, and this method is rapidly adopted by financial institutions to correlate associations between multiple securities. (David X. Li, 2012)

Different from standard credit derivative products, which are simply based on a single underlying credit risk, CDO is associated with a portfolio of credit risk, and therefore needs a distinct approach to evaluate default correlation. Tradition way to define default correlation is based discrete events, which categorize according to survival or nonsurvival at an important period such one year. For example, if we denote

$$q_A = Pr[E_A], \qquad q_B = Pr[E_B], \qquad q_{AB} = Pr[E_A E_B]$$

where E_A, E_B are defined as the default events of two securities A and B over 1 year. Then the default correlation ρ between two default events E_A and E_B, based on the standard definition of correlation of two random variables, are defined as follows

$$\rho = \frac{q_{AB} - q_A \cdot q_B}{\sqrt{q_A(1-q_A)q_B(1-q_B)}}$$

This discrete event approach has been taken by Lucas [1995]. Hereafter we simply call this definition of default correlation the discrete default correlation.

One disadvantage of this existing definition is its assumption that default correlation depends on a specific rather than a general time interval. In order to generalize the definition, we introduce a random variable called "time-until-default" to represent the length of time before a define point of event, usually known as default, is

occurring.

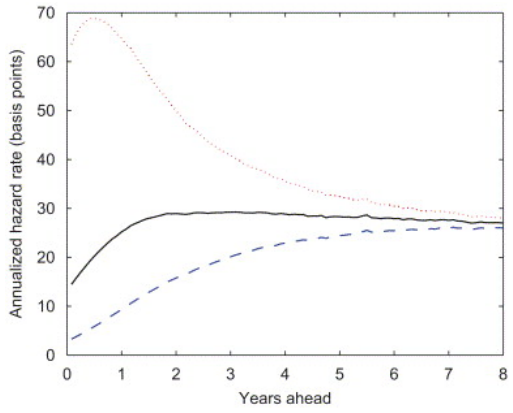
With this new random variable, we could define the default correlation of two entities A and B with respect to their survival times, or time-until-default, T_A and T_B as follows

$$\rho = \frac{Cov(T_A, T_B)}{\sqrt{Var(T_A)Var(T_B)}} = \frac{E(T_A T_B) - E(T_A)E(T_B)}{\sqrt{Var(T_A)Var(T_B)}}$$

Here is the marginal default probability of T_A and T_B up to any default time t in the future could be obtained by

$${}_t q_0 = Pr[\tau < t] = 1 - e^{-\int_0^t h(s)ds}$$

based on the corresponding credit curve,



This expression of default correlation is usually known as survival time correlation, which enhances the discrete default correlation by generalizing the dependent time interval. Now the question is, for an n credit portfolio, such as CDO, how should we

determine the joint distribution function given the marginal distribution and a correlation structure? The solution offered by the author is copula function.

In the two dimension Gaussian copula, which is also called bivariate normal copula

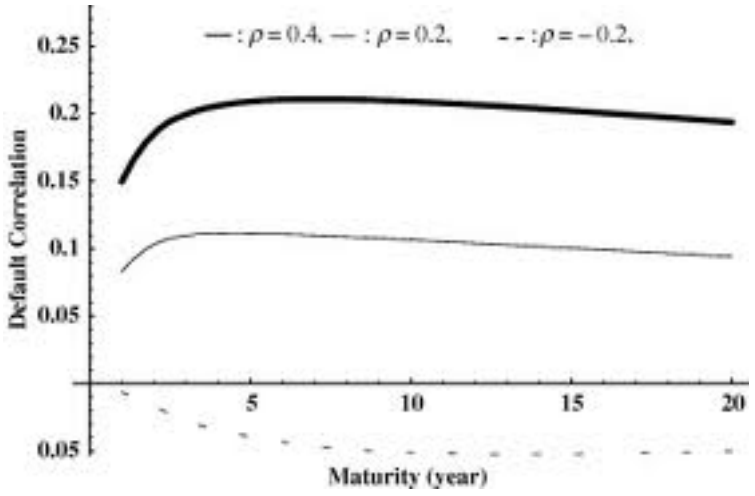
$$C(u,v) = \Phi_2(\Phi^{-1}(u), \Phi^{-1}(v), \rho), \qquad -1 \leq \rho \leq 1$$

where Φ_2 is the bivariate normal distribution function with correlation coefficient ρ, and Φ^{-1} is the inverse of a univariate normal distribution function. If we set the correlation parameter ρ to asset correlation and denote the survival times for A and B as T_A and T_B, the joint default probability can be calculated as follows (Li, 2000)

$$Pr[T_A < 1, T_B < 1] = \Phi_2(\Phi^{-1}(F_A(1)), \Phi^{-1}(F_B(1)), \gamma)$$

where F_A and F_B are the distribution functions for the survival times T_A and T_B.

Here is a sample default correlation versus length of time interval generated by applying the Gaussian copula.

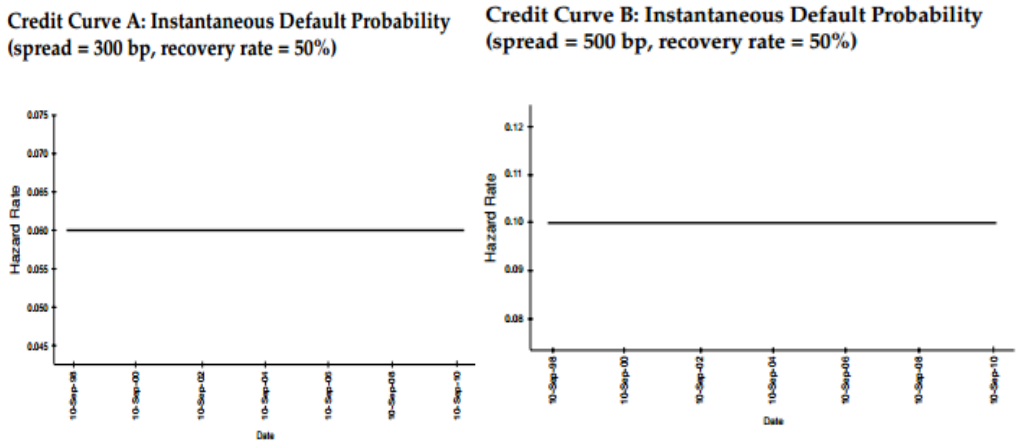


Deductively, we could use the same approach to construct high dimension copula to

model the credit portfolio of arbitrary size.

Here we provide a numerical example to illustrate the process of applying Gaussian copula to model default correlation versus time until default.

Given two credit curves as following:

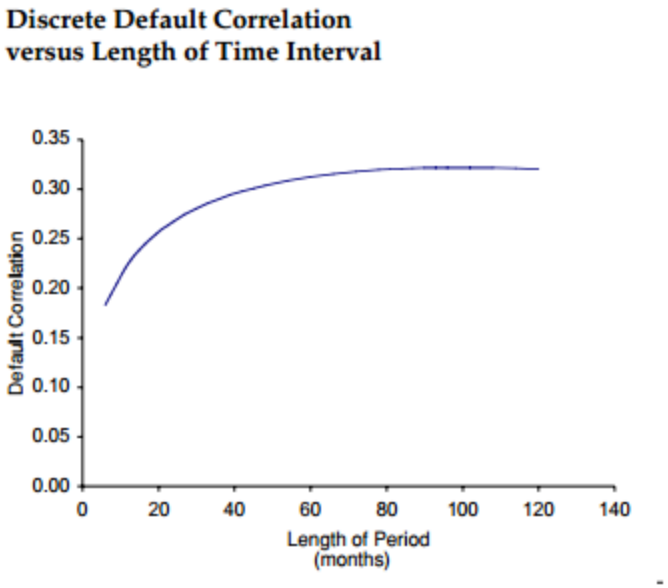


We then apply formula

$$ {}_tq_0 = Pr[\tau < t] = 1 - e^{-\int_0^t h(s)ds} $$

to obtain marginal default probability of T_A and T_B. Thusly the joint default probability can be calibrated by

$$ Pr[T_A < 1, T_B < 1] = \Phi_2(\Phi^{-1}(F_A(1)), \Phi^{-1}(F_B(1)), \gamma) $$



2.5.3 Random Factor Loading

Traders use CDO2 Calibration Tools to price the CDO by sending market data to the risk engine, which is implemented based on a mathematic model that extends the classical Gaussian Copula by introducing random factor loading.

The idea of random factor loading is published in "Extensions to the Gaussian copula: random recovery and random factor loadings" as a research result by Leif Anderson and Jakob Sidenius. (Leif Andersen, 2004)

In this extension, some shortcomings of the basic Gaussian model are well resolved. As all the portfolio credit models aim to do, RFL intends to stimulate the default co-dependence between different obligators. To put it simply, we would like to know whether a default obligator would likely to make another obligator susceptible to default. It is widely assumed by financial industry professionals that there exists the correlation between the defaults of each obligator, as is firmly backed up by the empirical study. To put it more specifically, the family of Gaussian Copula Modeling involves the usage of a copula function, of which use is to stitch together marginal single-obligator

default probabilities into a joint default distribution. (Leif Andersen, 2004) But same as its ancestor, RFL is still not and intends not to be a perfect economic model. People can enjoy the statistical convenience that gives them the approximation of a complex relation fairly easily, but totally relying on RFL proves to be disastrous, as what has already been shown in the financial crisis.

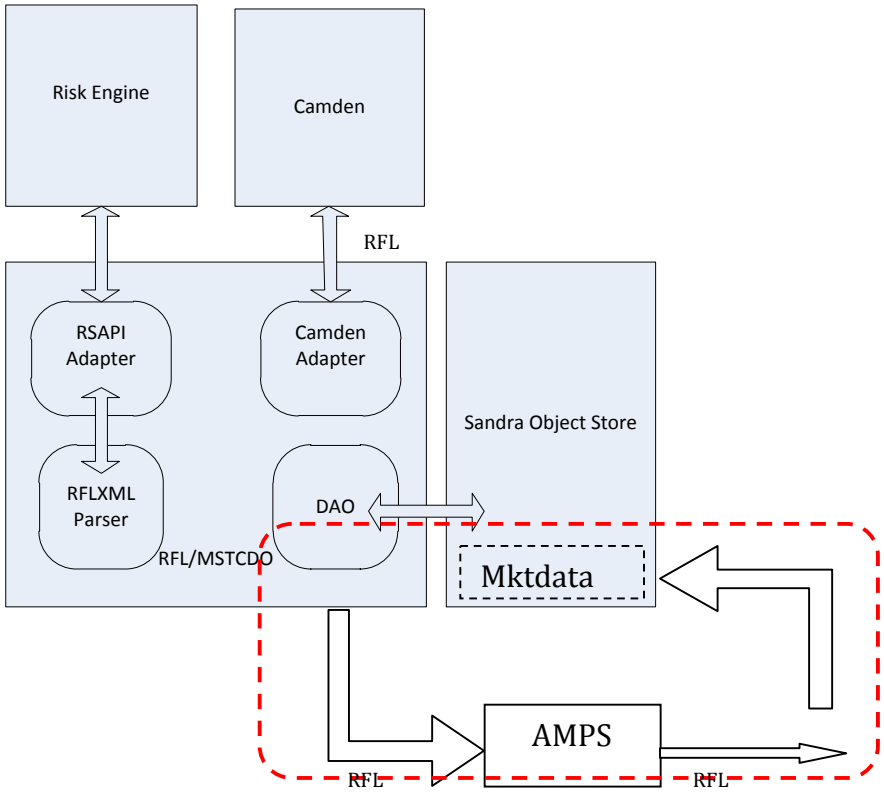
Chapter Three: Methodology

3.1 Saving RFL Factors to Sandra

"CDO^2 Calibration Tools" interfaces with Risk Engine directly and generate RFL calibration parameters for a given set of CDO^2 child trades and these results then be uploaded to Camden.

The current application is written in Python on the Quartz platform and the calculated results (RFL factors - Scale Result 1, Scale Result 2, Scale Result 3, Threshold Result 1 and Threshold Result 2) are uploaded to Camden.

Figure 1 below depicts a very high level system level overview and interactions of project 1 with the rest of system.



Project 1) Save RFL factors to Sandra

Figure 2: High Level System Overview of RFL Calibration Tool (1)

3.1.1 Market Model Object

In order to save RFL factors to Sandra, we need to decide how to store those data into a useful form. We need to define market data object. Currently, there are a total of seven values populated to the user interface:

Include: indicate whether a row of data is selected or not

Is Shadow: indicate if the row selected should be stored as the shadow set

Basket Name: the market basket name

Trade ID: each trade is identified with its own ID

Curve Name: name of the market curve

Detach Points: indicator for the measurement of seniority of CDO tranches

Initial Guess Key: the value of first key for the iteration procedure

After we submit the set of data above to the risk engine, we would expect to receive a set of three scale results and two threshold results upon successful processing of data by risk engine. The calculation results will be stored in the four arrays in the market data object as “ThresholdsBase”, “ThresholdsShadow”, “ScalesBase”, “ScalesShadow” with the corresponding market basket name and calibration parameters.

3.1.2 UI Update and Implementation of Related Logic

For the fact that CDO2 will upload RFL factors to Sandra along with saving them in Camden. Current UI needs to be updated and related logic needs to be implemented.

For example, user should be able to set environment variables such as MarketDataSet and MarketDataDate of Sandra database.

The current user interface only enables the user to configure the market data date setting. As a result, we also need to integrate within the interface the functionality to configure the market data set. In the process of saving RFL factor into Sandra database, an absolute path needs to be specified that indicates the directory where the factors should be stored. Part of the path will be "Mktdata" as illustrated by Figure2, under the "Mktdata" directory we need to append the market data set and market data date settings accordingly. The final directory path for writing RFL object will appear in the format "SandraDB/Mktdata/MarketDataSet/MarketDataDate/RFLObject".

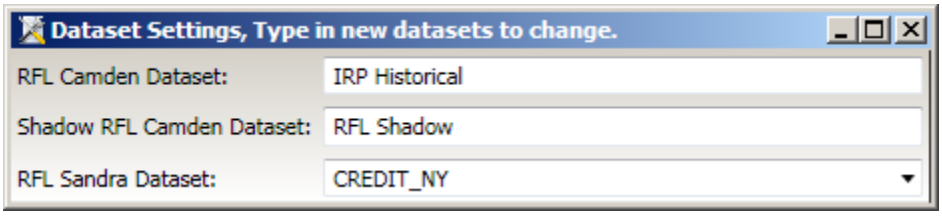


Figure 3: MarketDataSet Settings in CDO^2 Calibration/Marking Tool

In addition, parts of the current UI have been updated to make sure it is consistent with up-to-date convention.

1. Database environment information has been added to title bar.

2. Combine the date picker so that the new 'Database Publish Date' reflects the MarketDataDate for both Camden and Sandra database.

3. A new status bar has been added at the bottom to reflect current MarketDataSet and MarketDataDate of Sandra.

3.1.3 Save RFL Factors to Different Datasets

Currently, RFL factors are uploaded in Camden to two datasets, “IRP Historical” and “RFL Shadow”. We also need to consider this dataset concept and save RFL factors to different datasets in Sandra depending on “Is Shadow?” value on the UI. After our team design the new market data object which distinguishes the shadow and non-shadow set of scale results and threshold results by allocating four arrays, we eliminate the need to store the object in two different data sets in Sandra. Eventually, the shadow and non-shadow row of data will be condensed as one row and then written into the Sandra database.

During the first two weeks, our team has implemented the function that stores RFL object into Sandra by using AMPS and Bob scheduler in the Quartz platform. After constructing an AMPS message that contains all the necessary information about a RFL object, we then publish the message to the AMPS server “sct_dev” under the topic “SCT/RFL/PUB/REQUEST”. Finally, the Bob job script that listens to this topic writes the RFL object into Sandra database under the directory specified by the information contained in AMPS message.

However, one of the biggest disadvantages to store the RFL object via AMPS/Bob scheduler is its limited ability of error handling. Therefore, our team re-develops the function by using Hugs scheduler in the Quartz platform.

Hugs is a custom grid scheduler used by Quartz. It is implemented in C++ and currently runs on top of Data Synapse, but can also run independently. Hugs allows you to write Python code that runs in parallel on a distributed grid, which is why it is faster

than AMPS when we have a huge amount of data to send. It is kind of running tasks on distributed systems. Jobs can be monitored using the Hugs Monitor. (Hugs - The Quartz Grid Scheduler, 2012)

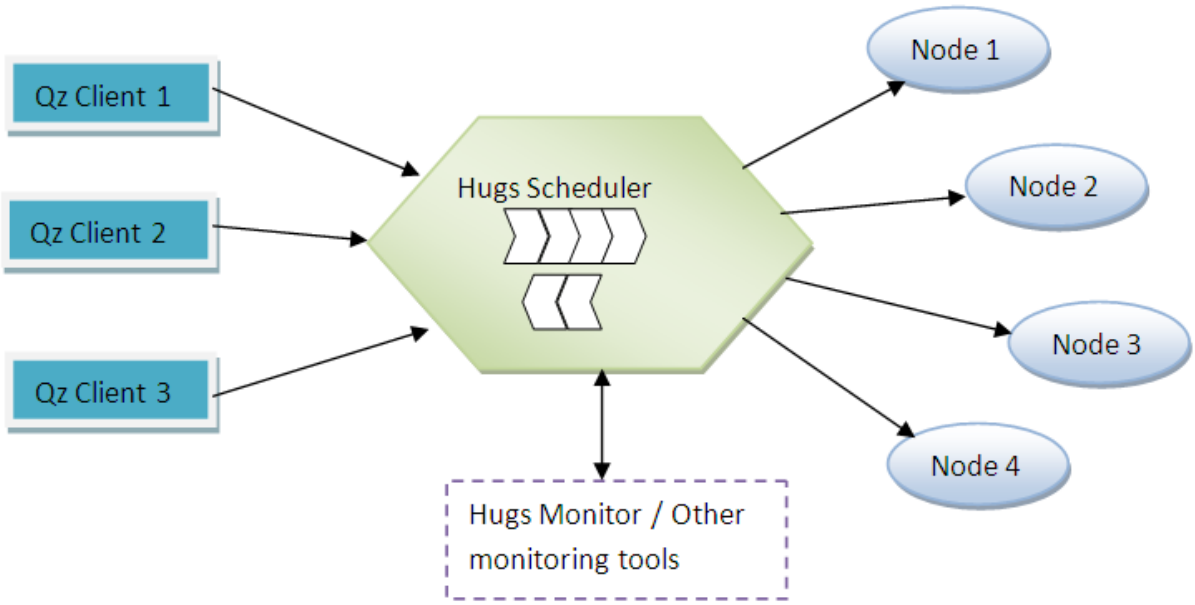
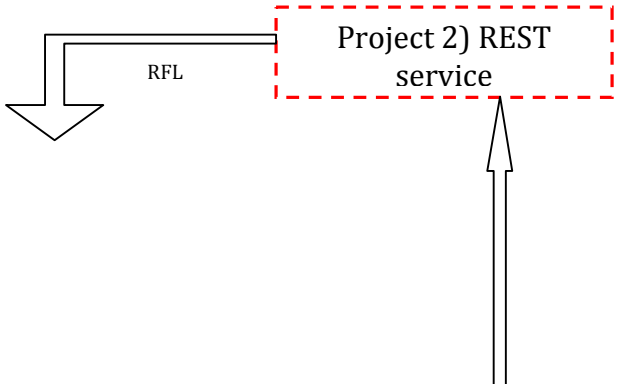


Figure 4: Hugs Components

3.2 Implementing REST service

This project's objective is to implement API Risk Engine will utilize to access published RFL factors that have been stored in Sandra.

Figure 2 below depicts a very high level system level overview and interactions of project 2 and the rest of the system.



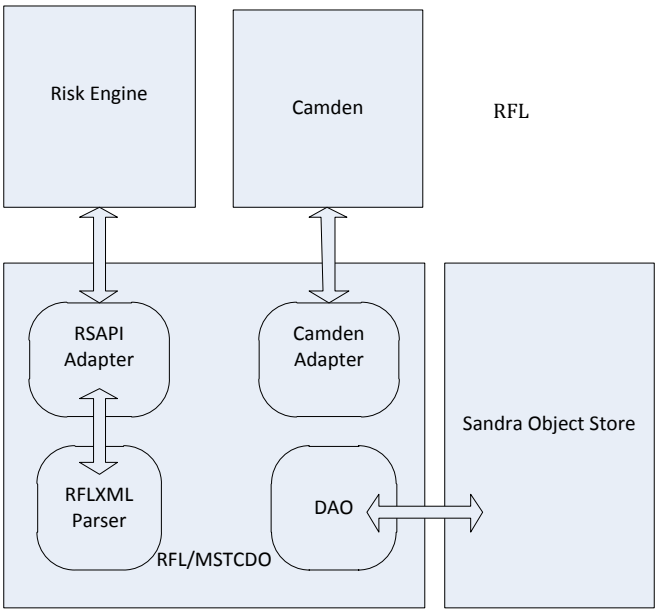


Figure 5: High Level System Overview of RFL Calibration Tool (2)

The REST Service will live on the Quartz grid and should be found using Quartz's Service Discovery protocol. The Service Discovery Protocol will provide the server name and port, which can be used to access the REST Service. Requests may be submitted to the service discovery server and will be automatically redirected to the RFL factors service.

3.2.1 Protocols for Requesting RFL Factors

It is the feature of REST service that it only makes use of the basic command, such as GET, POST, PUT, DELETE and other existing functionalities of the well-known HTTP protocol. The REST service greatly simplify the process of requesting RFL factors from Sandra database, as well as prevent the access issues which may cause problems for clients.

With the service running, a client will only need a URL to gain read access to the requested RFL Factors.

For Example:

- **GET /RFLFactors/{DataSet}/{MarketDate} HTTP/1.1**
    - Returns an array containing all RFL factor objects for dataset {DataSet} and market date {MarketDate}
- **GET /RFLFactors/{DataSet}/{MarketDate}/{Name} HTTP/1.1**
    - Returns the RFL factor object for the specified {DataSet} , {MarketDate}, and {Name}.

3.2.2 Register Quartz Discovery Service/Redirection

The RFL factor REST service is required to be registered within Quartz service directory, for the reason that any client looking to use the REST service could find it using the Quartz Discovery Service. To be specific, service clients can access all rest services through the discovery service URL. Requests will be automatically redirected to the corresponding service under the registered service name. It is worth noting that the URL will change every day, thusly making Quartz Discovery Service the only way through which we can locate the REST service.

3.2.3 JSON Object Formats

For the sake of convenience, it is required that all returned RFL Factors should be in the format of JSON. JSON, which is the abbreviation of JavaScript Object Notation, is a text-based open standard designed for human-readable data interchange. It is used primarily to transmit data between a server and web application, serving as an alternative to XML.

Example:

RFLFactor JSON Object

Name	Value
Name	String, corresponds to the identifier of the RFL factor

Entry	String, corresponds to market data entry date (i.e. market date) of the credit RFL factor, with format YYYYMMDD.
DataSet	String, corresponds to the (Sandra) dataset of the credit RFL factor.
QuotedBy	String, corresponds to the identity of the user or service which published the creditRFLFactor
QuotedAt	String, corresponds to the date and time the surface was quoted at in ISO8601 format (UTC time), e.g. "2012-11-30T16:51:14Z"
ScaleBase	Array of string, corresponds to value of scale in RFL base factor.
ScaleShadow	Array of string, corresponds to value of scale in RFL shadow factor.
ThresholdsBase	Array of string, corresponds to value of threshold in RFL base factor.
ThresholdsShadow	Array of string, corresponds to value of threshold in RFL shadow factor.

Table 2: RFLFactor Object Members

```
{
        'name': 'ML_DECIBEL_8654_PARENT',
        'entry': '2012-11-30',
        'dataSet': 'CREDIT_NY',
        'quotedBy': 'luyang.zhang',
        'quotedAt': '2012-11-30T16:51:14Z',
        'ScaleBase': ['4.454431266528465', '0.13498698341468446',
'0.1366913808148539'],
        'ScalesShadow': ['4.454509804485647', '0.1345869487137669',
'0.13635125877537008'],
        'ThresholdsBase': ['-4.0', '2.644373074971677E-4'],
        'ThresholdsShadow': ['-4.0', '4.254511530306401E-4'],
        'CalibrationParams': {}
}
```

Table 3: RFLFactor Object

Chapter Four: Analysis & Performance

4.1 Test Overview

In order to verify that an application operates as it was designed to, various types of test need to be conducted before the application gets pushed into production environment. Common types of tests include: unit tests that are run by developers, scenario tests that define an end to end business scenario, benchmark tests that validate the response times of the applications or scenarios, regression tests that extend scenario tests for all possible data points and are executed to ensure old functionality works in the wake of new changes, and performance tests that are a special form of regression tests that help validate the performance of the applications under varying stress/load conditions.

4.2 Unit Test for CDO2 Calibration/Marking Tool

The whole purpose of the project is to implement the functionality that enables CDO2 Calibration Tool to effectively communicate with Sandra database, as Bank of America are planning to remove Camden database completely next year and thusly facing the need to preserve the original database functionalities.

Inevitably and predictably, most of the quality assurance testing will focus on checking the consistency of performance when CDO2 Calibration Tool interacts with both databases.

Test Case No.	Publish to Camden	Publish to Sandra	Test case	Result
1	O	O	Camden publish and Sandra publish	Should both successful
2	O	X	Camden publish and Sandra publish	Camden: Rollback Sandra : Should not publish anything
3	X	O	Camden publish and Sandra publish	Camden: Should not publish anything Sandra : Rollback
4	X	X	Camden publish and Sandra publish	Camden: Should not publish anything Sandra : Should not publish anything

Table 4: Unit Test Cases for CDO^2 Calibration/Marking Tool

The table above illustrate all the possible outcomes of CDO2 Calibration Tool publish function. The current publish function will write the data to both Camden and Sandra database and therefore introduces four situations that could possibly happen. Test case1 would be the ideal situation, which neither Camden nor Sandra encounters any problems when CDO2 Calibration Tool writes the RFL object into the database. Under this circumstance, we need to compare the objects that are finally saved in both databases.

However, in test case2, only Camden successfully receives the data published from the user interface. In this case, we need to rollback the Camden publish process and log any errors or exceptions raised while the CDO2 Calibration Tool is trying to publish to Sandra database. Because of the implementation of publish function, publish to Sandra will not be attempted if publish to Camden is not successful. The logic of implementation eliminates the possibility of test case3.

Lastly, it is obvious that nothing should be written to both databases, if neither one of the publish attempts succeeds.

Chapter Five: Conclusion

The previous CDO^2 Calibration/Marking Tool interfaces with the risk engines and saves all the calculation results into Camden database, which is a traditional SQL database. However, for a risk & trading system where users generally end up storing complex instruments, model parameters, trades, and market data objects, SQL database has the disadvantage of potentially exposing the internal data representation to end users. Although stored procedures and views could in some extent alleviate this situation, SQL doesn't enforce this practice and makes it easier for the schemas to leak out into interface.

Sandra, as an object database, is thusly used by all regular Quartz applications. It is written in C++ for performance reasons, but its main API is in Python. By creating a general purpose object database that is closely integrated with the pricing/risk framework, we can simplify the task when we need to define new instruments, pricing models, or changes the attributes on the existing objects.

In this project, we have accomplished the database migration from Camden to Sandra by implementing the related database functionalities by using Hugs grid scheduler. The function enables the CDO^2 Calibration/Marking Tool to interface and publish data into Sandra database with basic error checking and reporting attributes. A complete and thorough QA test has been conducted by comparing and analyzing the performances and results generated when CDO^2 Calibration/Marking Tool interacted with both databases.

References & Glossary

References

Arbenz, P. (2011). "Bayesian Copulae Distributions, with Application to Operational Risk Management - Some Comments". *Methodology and Computing in Applied Probability Forthcoming* .

Bank of America Overview. (2012). Retrieved 2012 йил December from Bank of America: http://investor.bankofamerica.com/phoenix.zhtml?c=71595&p=irol-homeprofile#fbid=nEzwyDXx8OL

Bloomberg. (2010). CLOs to End 12-Month Drought in Citigroup Deal: Credit Markets. *Bloomberg* .

CDO-Squared. (2012). Retrieved 2012 йил December from Investopedia: http://www.investopedia.com/terms/c/cdo2.asp#axzz2EHpMMYQH

Copula (probability theory). (2012 йил October). Retrieved 2012 йил December from wikipedia: http://en.wikipedia.org/wiki/Copula_(probability_theory)

David X. Li. (2012). Retrieved 2012 йил December from wikipedia: http://en.wikipedia.org/wiki/David_X._Li

Global Markets and Risk Technology (GMRT). (2012). Retrieved 2012 йил Dec from Bank of America: http://campus.bankofamerica.com/americas/analyst/technology-developer-and-analyst-program-%E2%80%93-global-markets-and-risk-technology-(gmrt).aspx

Hugs - The Quartz Grid Scheduler. (2012). Retrieved 2012 йил December from Quartz Documentation:

http://lnyce23217.bankofamerica.com:8181/docs/sphinx/html/components/hugs.html

Katherine, A. (2009). The Story of the CDO Market Meltdown.

Leif Andersen, J. S. (2004). *Extensions to the Gaussian copula.* Retrieved 2012 йил December from Journal of Credit Risk: http://www.risk.net/digital_assets/4487/v1n1_Andersen_new.pdf

Li, D. X. (2000). On Default Correlation: A Copula Function Approach. *Journal of Fixed Income* , 43-54.

Quartz Academy - Overview Session. (2010). Retrieved 2012 йил December from Quartz Documentation: Bank of America internal web page

QzDesktop . (2010). Retrieved 2012 йил December from Quartz Documentation: Bank of America internal web page

Relevant Quartz Components. (2010). Retrieved 2012 йил December from Quartz Documentation: Bank of America internal web page

Sandra Features. (2010). Retrieved 2012 йил December from Quartz Documentation: Bank of America internal web page

Schmidt, T. (2006). *Coping with Copulas.* Leipzig: Copulas - From Theory to Applications in Finance.

Structured credit. (2012). Retrieved 2012 йил December from Creditflux: http://www.creditflux.com/Glossary/Structured-credit/

Glossary

AMPS: AMPS (Advanced Message Processing System) is a high performance publish-subscribe messaging system, with database-like, State-of-the-World (SOW) querying functionality.

BOB: Bob is the scheduling system for Quartz. You can run a Bob agent on any Linux EFS-enabled machine to add your machine as a slave to run jobs on.

CDO: Collateralized debt obligations (CDOs) are a type of structured asset-backed security (ABS) with multiple "tranches" that are issued by special purpose entities and collateralized by debt obligations including bonds and loans.

Hugs: Hugs is a custom grid scheduler used by Quartz. It is implemented in C++ and currently runs on top of Data Synapse, but can also run independently. Hugs allows you to write Python code that runs in parallel on a distributed grid. Jobs can be monitored using the Hugs Monitor. Hugs history can be browsed using Hugs History.

Qztable: A qztable.Table represents a tabular data set or a timeseries in quartz. Qztable can load data from a range of sources: sql, csv, hdf5, kdb, etc. as well as programmatically in Python. Qztable is written in C++, and interfaces to quartz through swig.

REST: REpresentational State Transfer (REST) is a style of software architecture for distributed systems such as the World Wide Web. REST has emerged as a predominant Web service design model.

Sandra: Sandra is the object database used by all regular Quartz applications. It is written in C++ for performance reasons, but its main API is in Python.

YAML: YAML is a human-readable data serialization format that takes concepts from programming languages such as C, Perl, and Python, and ideas from XML and the data format of electronic mail (RFC 2822). YAML was first proposed by Clark Evans in 2001, who designed it together with Ingy döt Net and Oren Ben-Kiki. It is available for several programming languages.

Appendix A: Quartz Development Process

ANALYSIS AND DESIGN
Role • Developer
Prerequisites •
Description • Define requirements • Define architecture
Meeting Sessions • Persons responsible for QA and Dev make sure the knowledge of functionality is handed over
Check Points for Successful Completion • JIRA exists

DEVELOPMENT
Role • Developer
Prerequisites • JIRA exists
Description • Develop functionality • Develop tests • Peer review code
Meeting Sessions • Dev informs QA when the functionality is tied down enough for QA to implement tests • PM and Dev ensures that development will complete before intended QA Testing cycle

Check Points for Successful Completion
• Code is reviewed and approved
• Tests have acceptable coverage, including
  ○ Testing intended function
  ○ Testing integration with neighbouring components
  ○ Testing special cases
• All tests run successfully
• JIRA handed over to QA

ANALYSIS AND DEVELOPMENT
Role

- QA

Prerequisites

- JIRA Exists
- Requirements and functionality well known by QA.

Meeting Sessions

- Persons responsible for QA and Dev make sure the knowledge of functionality is handed over
- Dev and QA teams discuss to identify functionality potentially affected by the new functionality

Description

- Define QA Tests for new functionality
- Identify existing functionality which will be affected
    - Identify existing QA tests for affected functionality **or**
    - Develop new QA tests for testing interaction between the new and existing functionality.
- Automate QA Tests
- Update Regression suite
    - Select tests from Automated QA Test suite as well as Developers' tests.
- Review QA Test coverage

Check Points for Successful Completion

- QA Tests have acceptable coverage
- Affected functionality has been identified and QA Tests for their interaction with the new functionality are available.
- Regression suite is updated

QA TESTING

Role

- QA

Prerequisites

- JIRA handed over to QA
- Functionality has been developed
- QA Tests have been identified and developed

Meeting Sessions

- QA informs Dev of bugs
- Dev and QA teams discuss to identify functionality potentially affected by the new functionality

Description

- Run QA Tests
- Run Free-form testing
- Defer ticket to developer if bug is found
- Write JIRA tickets for non-related errors

Check Points for Successful Completion

- All active QA Tests run successfully
- Tickets for newly found bugs written

USER ACCEPTANCE TESTING
Role • UAT
Prerequisites • All active QA Tests run successfully
Description • User signs off new functionality • User signs off the removal of bugs
Check Points for Successful Completion • User has signed off all functionality in the ticket

PRODUCTION
Role • User
Prerequisites • User has signed off functionality in the ticket
Description -
Check Points for Successful Completion -

Appendix B: CDO2 Calibration Tool Test Process

1. Launch CDO2 Calibration Tool:

In order to test CDO2 Calibration Tool, please look for CDO2 Calibration Tool in QzDesktop. (/Credit/Beta/TEST/CDO2 Calibration Tools (QA))

2. How to run CDO2 Calibration Tool scripts from QzDev:

All relevant scripts were pushed to "sct_staging" area. The scripts pushed are as follows:

Script Name	**Is Main?**
credit/sct/apps/RFL/RFLSandraDAO.py	
credit/sct/apps/RFL/RFLPanel.py	**Yes**
credit/sct/apps/RFL/RFLXML.py	
credit/sct/apps/RFL/rflhugs.py	
credit/sct/apps/RFL/__init__.py	

To run the CDO2 Calibration Tool script, open "**credit/sct/apps/RFL/RFLPanel.py"** in QzDev.

3. How to check final results stored:

Check List before click the Publish button on CDO2 Calibration Tool:

1. Make sure Risk Engine Env = QA, Camden Env: QA

2. Selected the rows to publish and submit to Risk Engine

3. Wait until calculation results populate to the user interface (numbers should be populated in following five columns: Scale Result1, Scale Result2, Scale Result3, Threshold Result1, Threshold Result2)

4. Make sure dataset settings (settings -> Dataset Settings) are configured as following:

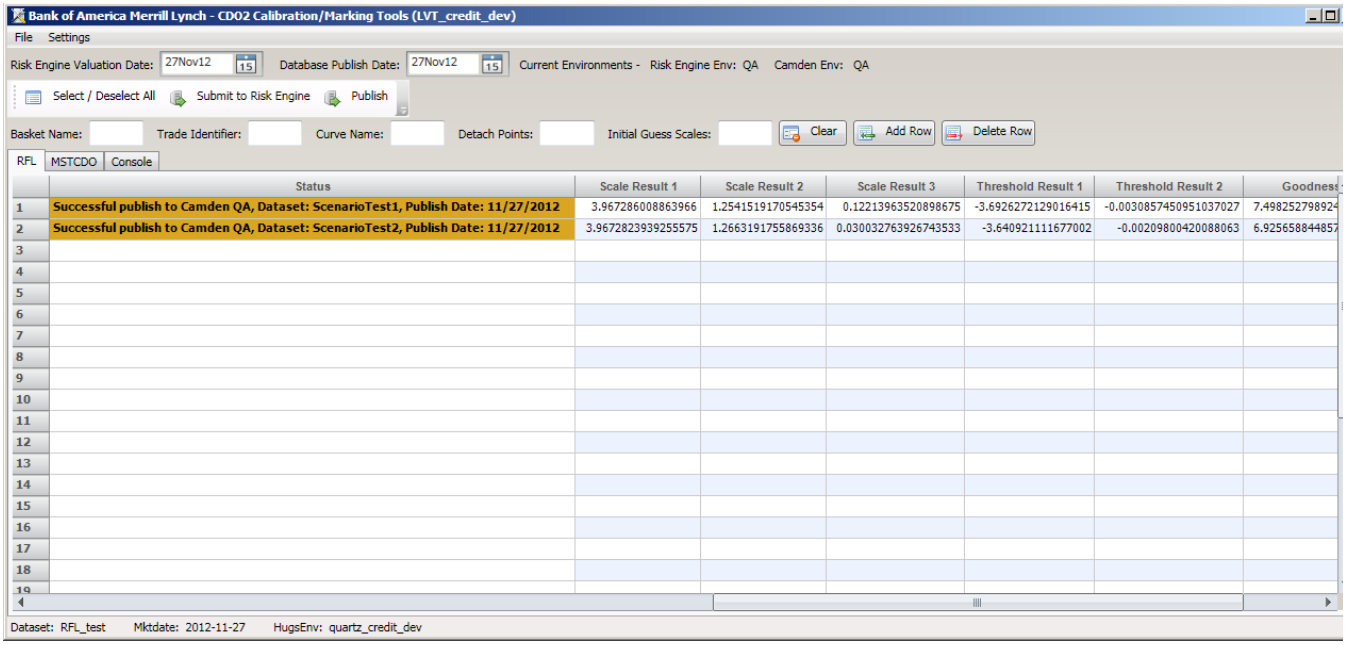
    a. RFL Camden Dataset: ScenarioTest1

    b. Shadow RFL Camden Dataset: ScenarioTest2

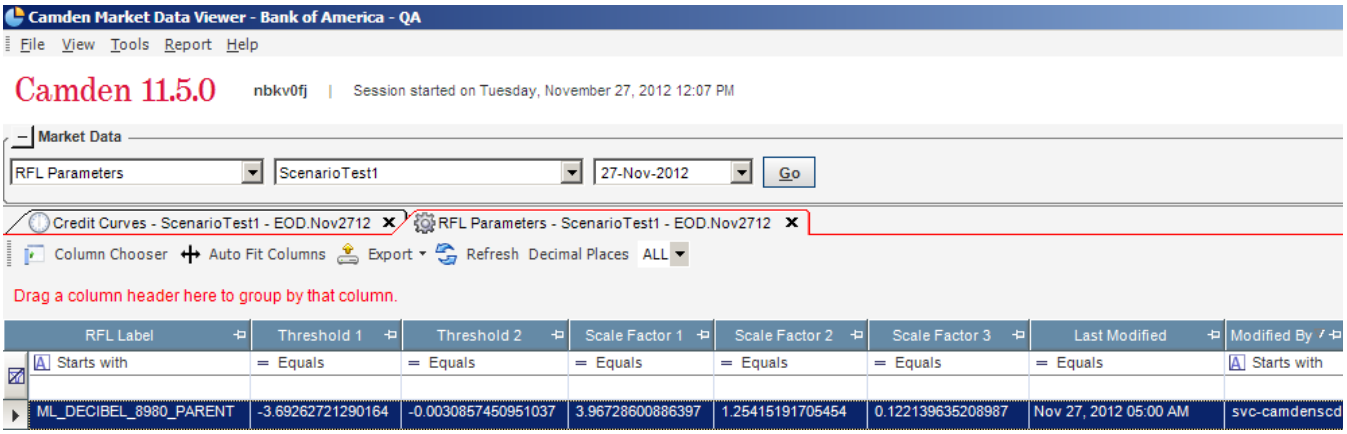
    c. RFL Sandra Dataset: RFL_test

5. Click Publish button

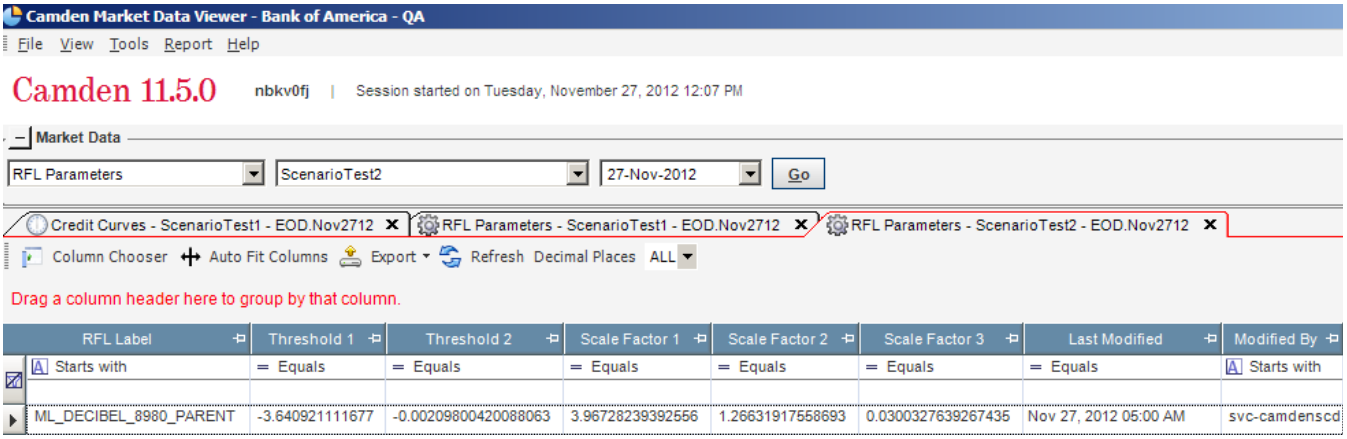
CDO2 Calibration Tool:



Scenario Test1 (Base Results): Check if calculated numbers were correctly published to Camden for the basket name (RFL Label). For testing purpose, "ScenarioTest1" Camden dataset is used to publish only **base** rows. (Rows on UI that "IsShadow" column NOT checked.)



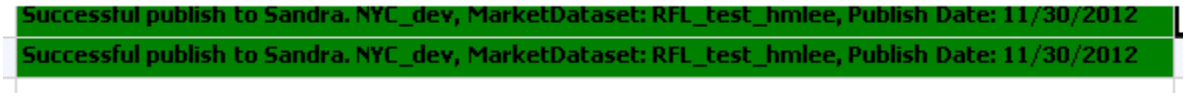
Scenario Test2 (Shadow Results): Check if calculated numbers were correctly published to Camden for the basket name (RFL Label). For testing purpose, "ScenarioTest2" Camden dataset is used to publish only shadow rows. (Rows on UI that "IsShadow" column **checked**.)



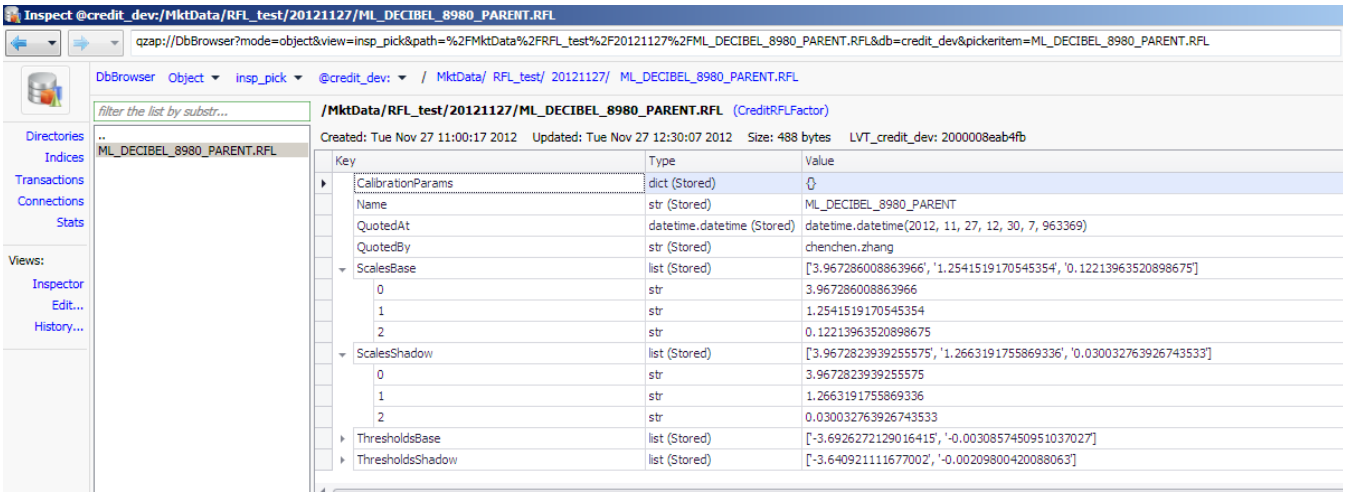
Sandra Database:

If selected rows were successfully published to Sandra, following message will be displayed on the row on UI:

Example: It shows which Sandra DB and MarketDataSet the RFL numbers published to.



Unlike in Camden which publishes Base and Shadow numbers in separate Camden dataset (i.e. ScenarioTest1 and ScenarioTest2), we save only **ONE** object for shadow and base rows with same basket name (which is RFL Label). This CreditRFLFactor Sandra object contains ScalesBase, ScalesShadow, ThresholdsBase, ThresholdsShadow fields and each calculated numbers in UI will be saved in its respective Sandra object field in lists.



4. Error testcase handling:

The tool will try to publish in the order of **1) first to Camden, and if this is done successfully then 2) to Sandra**.

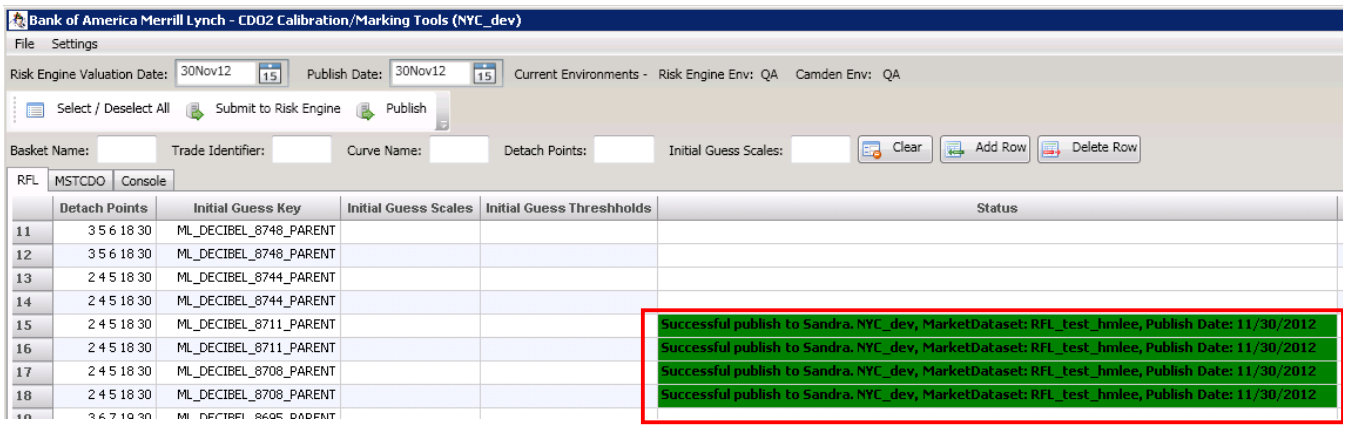
The behaviors that will happen in each possible error cases are listed below.

No.	Publish to Camden	Publish to Sandra	Test Case	Result
1	Successful	Successful	Camden publish and Sandra publish	Should both successful.
2	Successful	Error	Camden publish and Sandra publish	Camden: Should be published successfully. Sandra: Should **not** publish anything, error message will be displayed on the UI. This error occurred entries should be re-published by user manually from the UI.
3	Error	Successful	Camden publish and Sandra publish	Camden: Should not publish anything. Sandra: If Camden publish fails, the tool stops publishing and does not even attempt to publish in Sandra at all. These rows should be re-published by user manually from the UI.
4	Error	Error	Camden publish and Sandra publish	Camden: Should not publish anything. Sandra : Should not publish anything. These rows should be re-published by user manually from the UI.

In following section we explain each test case in more detail.

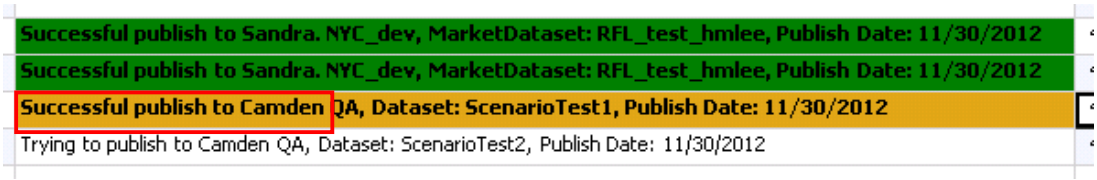
Test Case 1:

If everything goes well, that's what we expect to see!

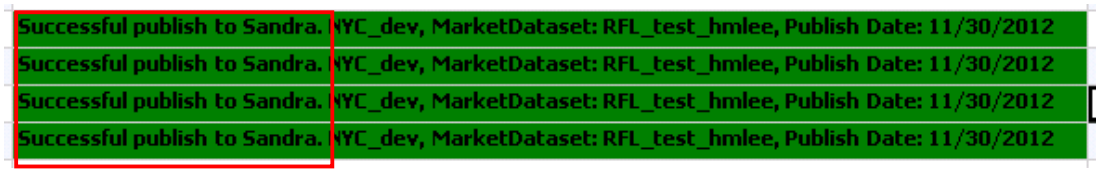


On UI:

1) It will publish first to **Camden** and then **Sandra**.



2) When publish is completed the UI will look like this:

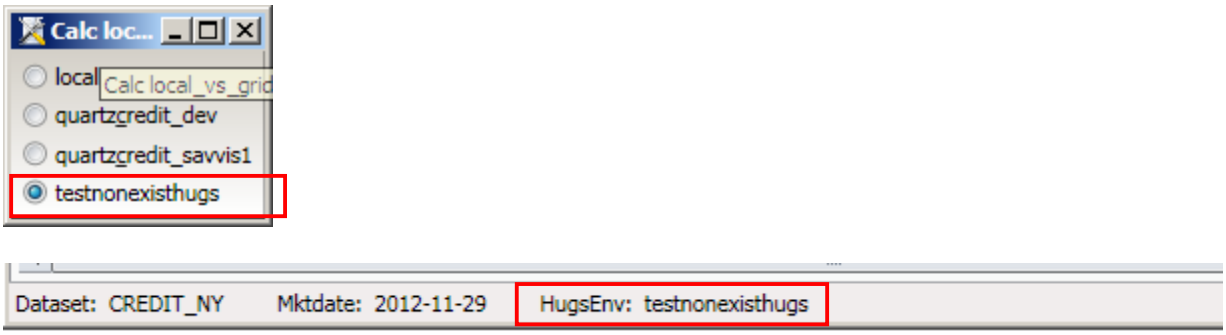


Test Case 2:

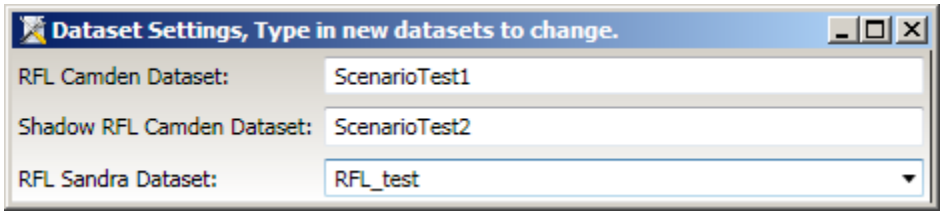
We want to test the case where RFL factors can be successfully published to Camden, while have problems during Sandra publish. One possible reason for failure in Sandra is that the Hugs environment specified in CDO^2 Calibration Tool can't be found.

Procedure:

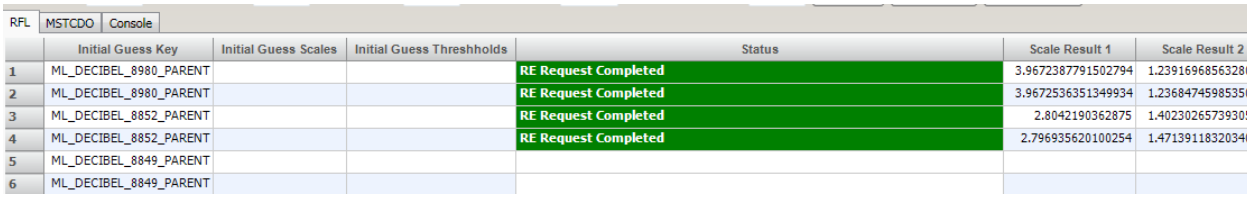
1. Choose a dummy Hugs scheduler in Settings -> Computer Grid Settings. This will cause publish to Sandra fails (while publish to Camden will be successful.).



2. Under Settings -> Dataset Settings, let RFL Camden dataset and Shadow RFL Camden dataset be 'ScenarioTest1' and 'ScenarioTest2', respectively. Double-check the data entries for corresponding date have been created so that publishing to Camden should be all smooth.



3. Submit RFL Factor Parameters to risk engine. Wait until all the calculation results are ready to send.



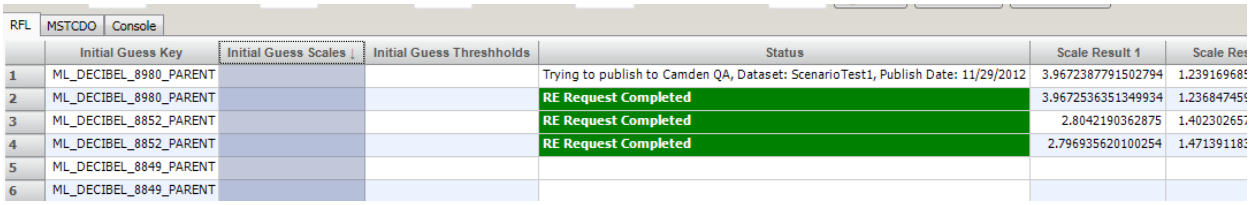
	Initial Guess Key	Initial Guess Scales	Initial Guess Threshholds	Status	Scale Result 1	Scale Result 2
1	ML_DECIBEL_8980_PARENT			RE Request Completed	3.9672387791502794	1.2391696856328
2	ML_DECIBEL_8980_PARENT			RE Request Completed	3.9672536351349934	1.2368474598535
3	ML_DECIBEL_8852_PARENT			RE Request Completed	2.8042190362875	1.4023026573930
4	ML_DECIBEL_8852_PARENT			RE Request Completed	2.796935620100254	1.4713911832034
5	ML_DECIBEL_8849_PARENT					
6	ML_DECIBEL_8849_PARENT					

In this case, we select RFL Factors ML_DECIBEL_8980_PARENT and ML_DECIBEL_8852_PARENT (referred to as 'M8980' and 'M8852' in the following text). If everything goes well, at last two base records and two shadow

records will be sent to ScenarioTest1 and ScenarioTest2 in Camden, respectively, and two RFL Factor objects will be sent to Sandra.

4. Click ‘Publish’.

   a) Start to send the base of ‘M8980’ to Camden.



RFL | MSTCDO | Console

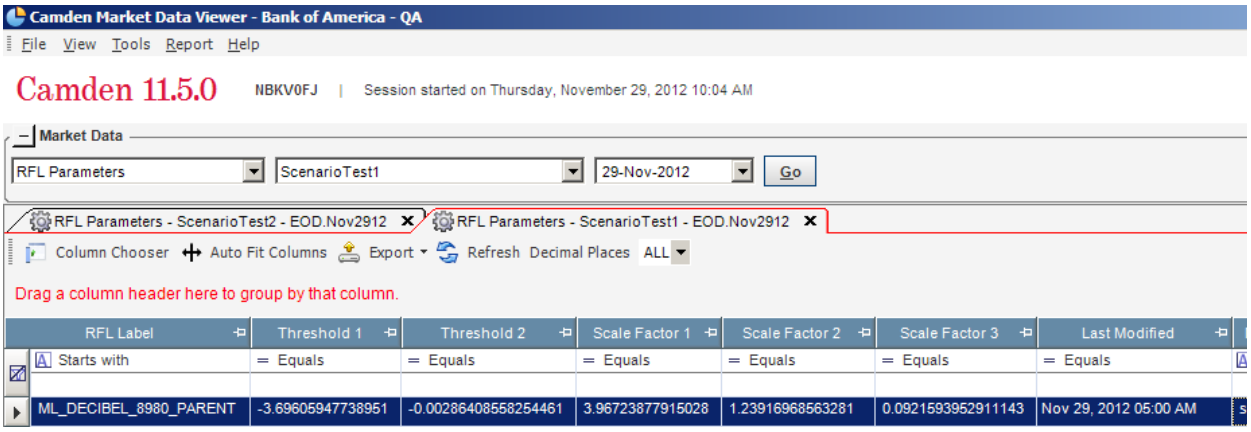
	Initial Guess Key	Initial Guess Scales	Initial Guess Threshholds	Status	Scale Result 1	Scale Res
1	ML_DECIBEL_8980_PARENT			Trying to publish to Camden QA, Dataset: ScenarioTest1, Publish Date: 11/29/2012	3.9672387791502794	1.239169685
2	ML_DECIBEL_8980_PARENT			RE Request Completed	3.9672536351349934	1.236847459
3	ML_DECIBEL_8852_PARENT			RE Request Completed	2.8042190362875	1.402302657
4	ML_DECIBEL_8852_PARENT			RE Request Completed	2.796935620100254	1.471391183
5	ML_DECIBEL_8849_PARENT					
6	ML_DECIBEL_8849_PARENT					

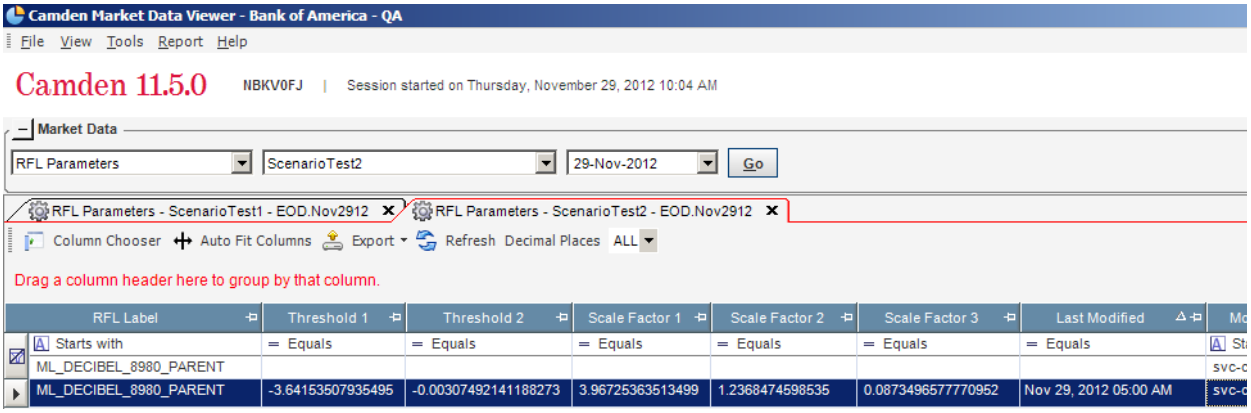
   b) Base of ‘M8980’ is successfully sent to ScenarioTest1 in Camden. Since we don’t have the complete RFL Factor object, we will wait until the shadow of ‘M8980’ is sent to Camden and start to publish it to Sandra if both records are successfully saved in Camden.

23	T_Global1 5YR	3 6 7 19 30	ML_DECIBEL_8672_PARENT			Successful publish to Camden QA, Dataset: ScenarioTest1, Publish Date: 11/30/2012
24	T_Global1 5YR	3 6 7 19 30	ML_DECIBEL_8672_PARENT			Trying to publish to Camden QA, Dataset: ScenarioTest2, Publish Date: 11/30/2012

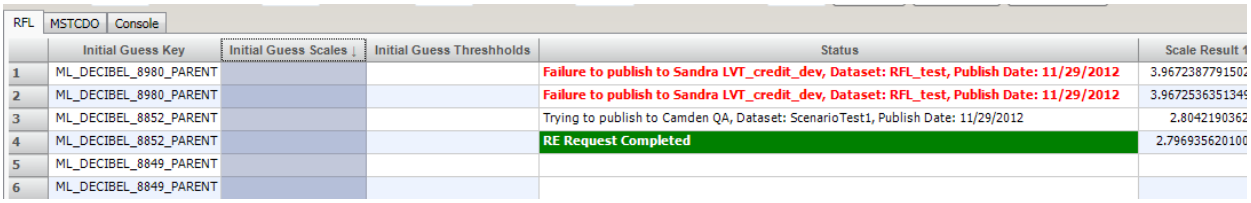
In above screen shot it shows publish to Camden has been successfully completed.

   c) After ‘M8980’ have been fully sent to Camden, we begin to try publishing ‘M8980’ to Sandra.

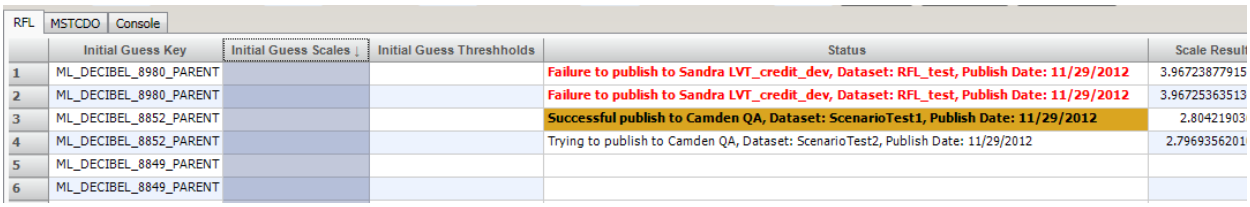




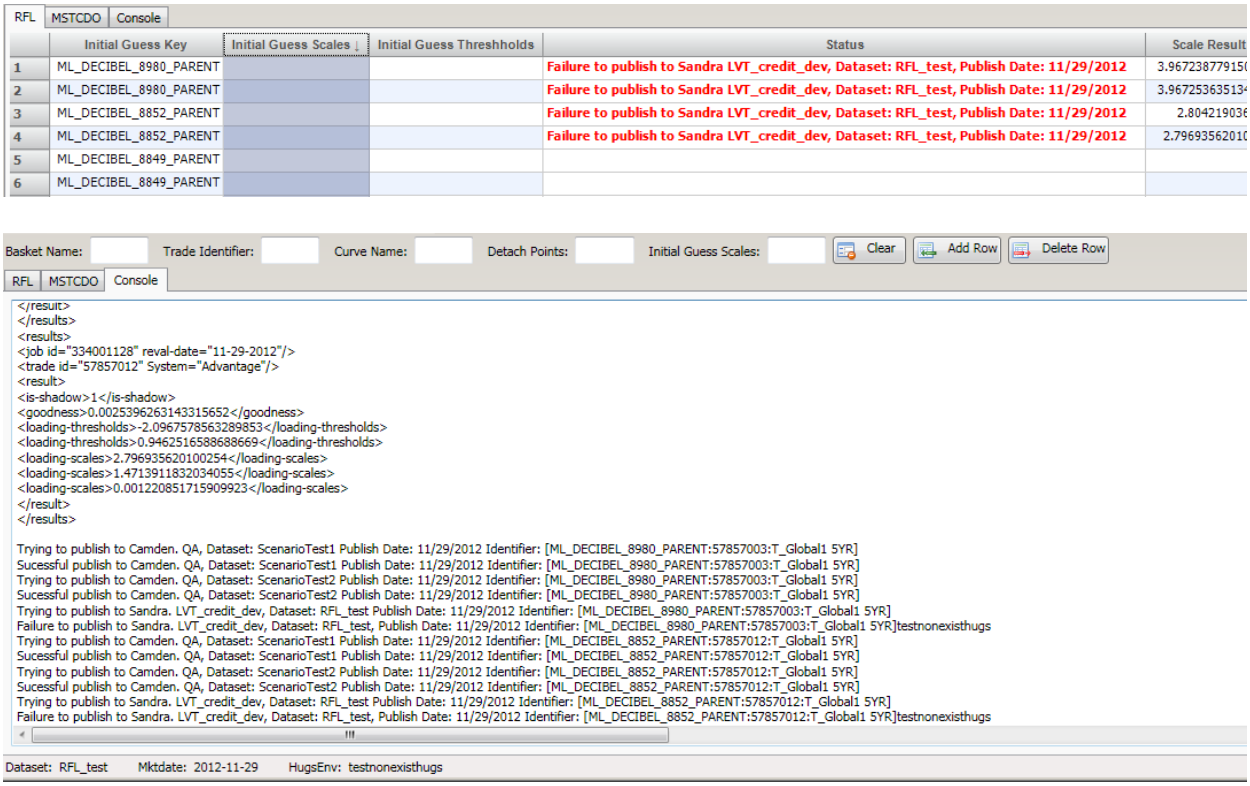
d) Failure to publish to **Sandra** because the Hugs scheduler we choose doesn't exist. Log the error in the console and continue with the next object, 'M8852'.



e) Basically 'M8852' proceeds the same way.



f) We failed all the Sandra jobs…In this case, the calculation results of 'M8890' and 'M8852' will remain in Camden. But error messages will show in the console saying Sandra object needs to be republished later. Nothing will be published for these entries in Sandra.



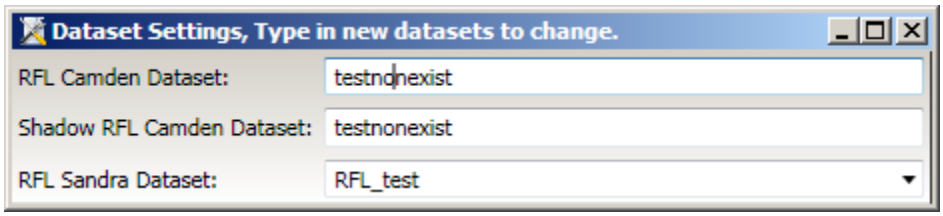
Test Case 3 & 4:

Test Case 3 and Test Case 4 are grouped together because they enjoy something in common. If any error occurs during the process of publishing base or shadow of RFL Factor to **Camden**, we will skip the part of publishing this object to Sandra.

To cause a failure in publishing to Camden, we can choose a dataset which doesn't exist in Camden.

There are two possibilities under this test case.

1. Both Camden publishes failed; In this case nothing will be published to both Camden and Sandra.



RFL | MSTCDO | Console

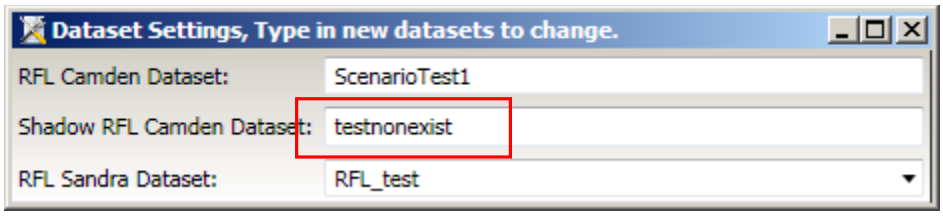
	Curve Name	Detach Points	Initial Guess Key	Initial Guess Scales	Initial Guess Threshholds	Status
1	T_Global1 5YR	2 4 5 18 30	ML_DECIBEL_8980_PARENT			Failure to publish to Camden QA, Dataset: testnonexist, Publish Date: 11/29/2
2	T_Global1 5YR	2 4 5 18 30	ML_DECIBEL_8980_PARENT			Failure to publish to Camden QA, Dataset: testnonexist, Publish Date: 11/29/2
3	T_Global1 5YR	3 6 7 19 30	ML_DECIBEL_8852_PARENT			Failure to publish to Camden QA, Dataset: testnonexist, Publish Date: 11/29/2
4	T_Global1 5YR	3 6 7 19 30	ML_DECIBEL_8852_PARENT			Failure to publish to Camden QA, Dataset: testnonexist, Publish Date: 11/29/2
5	T_Global1 5YR	3 6 7 19 30	ML_DECIBEL_8849_PARENT			
6	T_Global1 5YR	3 6 7 19 30	ML_DECIBEL_8849_PARENT			

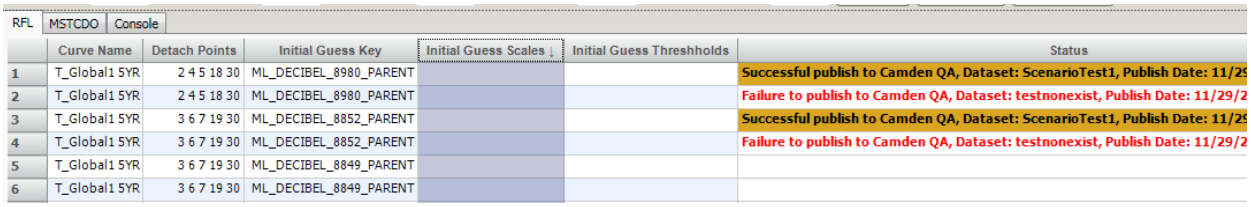
Errors message is logged in the console and no need to republish,

2. Either base or shadow failed in publishing.

   In this case, publish to Camden dataset ScenarioTest1 (for base) will be completed successfully. Publish to non-existing Camden dataset will cause error and in this case nothing will be published to Camden.

If any of the rows (shadow or base) for the same basket name occurred error, nothing will be published to Sandra for that basket name. Only the baskets that both shadow and base rows were successful will be published to Sandra. (This is because shadow and base rows for the same basket will be created as **ONE** Sandra object).





Errors message is logged in the console and the user will have to republish later.

Possible Improvements to do:

1. The error message of Sandra error is not very useful. It could be hard to tell from the error message where something goes wrong.

2. There is no permanent log. Currently we only record all error information in the console of the application, which will be lost if the app is closed.

3. If by any chance we have to republish the RFL Factor to Sandra, we couldn't skip the part of first republishing it to Camden, which seems like a waste of time and may be something what we want to avoid.

4. It is not so easy to see from the status column whether a particular RFL Factor object needs to be republished. (As far as I am concerned, only in the case that both Camden publishes failed, a republish will not be required. Otherwise we have to do the republish later to make sure Camden and Sandra are consistent.)

Appendix C: REST Service for CDO2 Calibration Tool Test Process

1. Launch REST service browser:

While the discovery service is generally intended for programmatic access, we can browse the available REST services here: http://qzsd.bankofamerica.com:8814/

Click into sct-rfl-rest-dev. You can find the URL of RFL REST service through the Quartz Discovery Service.

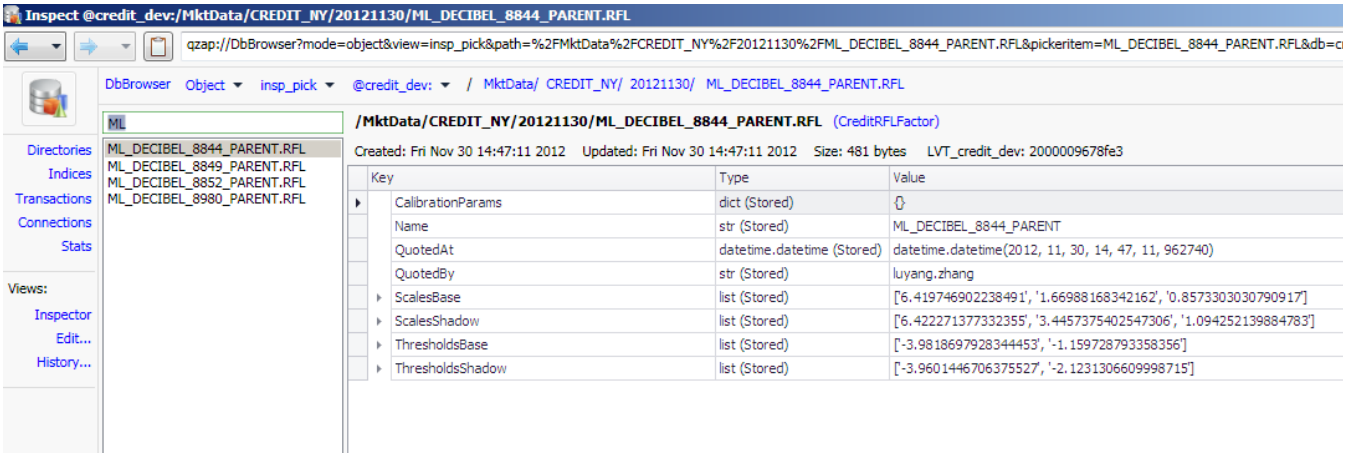
Click the URL. We entered the service routing mapo for RFL REST service. Currently there are two services provided to gain access to RFL factors in Sandra.

It can either return an array containing all RFL factor objects for given dataset{DataSet} and market date{MarketDate}, or return the RFL factor object for the specified dataset{DataSet}, market date{MarketDate} and name{Name}.

Service Routing Map		
HTTP methods	RESTer routing rule	Service Endpoint handle
GET	/ui	@web_ui
GET	/creditRFLFactors/<DataSet>/<MarketDate>/<Name>	credit_rfl_service.getCreditRFLFacotrByDataSet_MarketDate_Name
GET	/creditRFLFactors/<DataSet>/<MarketDate>	credit_rfl_service.getCreditRFLFactorByDataSet_MarketDate

2. How to check REST service read correct RFL factors

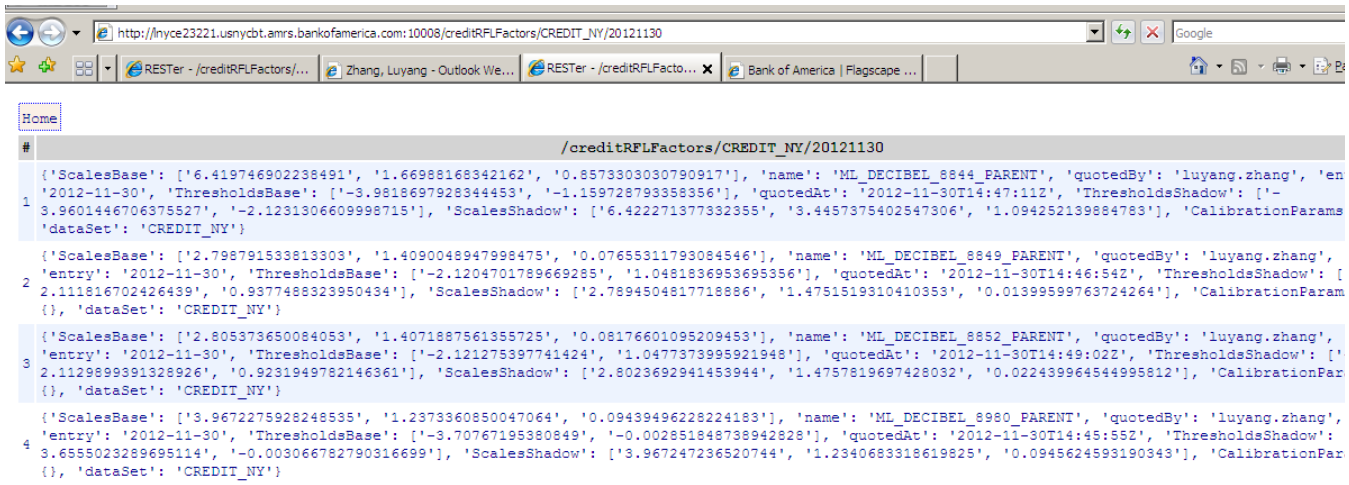
Open the Sandra DB browser. Switch to credit_dev database. Then go to the directory /MktData/CREDIT_NY/20121130/ and there are four RFL factors saved under that date.



Enter

http://lnyce23221.usnycbt.amrs.bankofamerica.com:10008/creditRFLFactors/CREDIT_NY/20121130

We get returned with the JSON representations of RFL objects under that date.

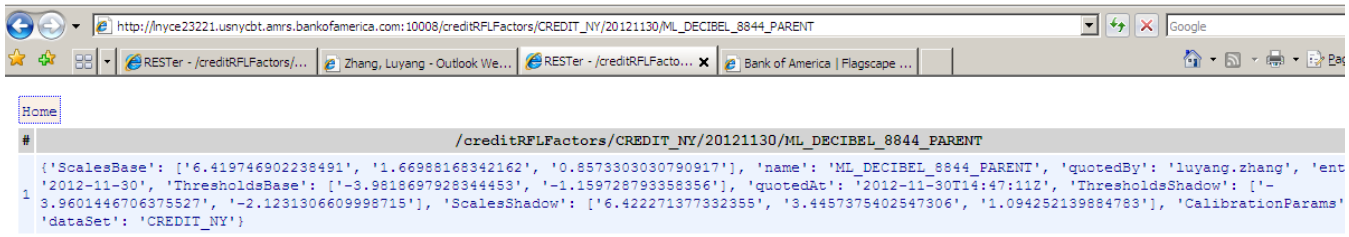


Enter

http://lnyce23221.usnycbt.amrs.bankofamerica.com:10008/creditRFLFactors/CREDIT_NY/20121130/ML_DECIBEL_8844_PARENT.

We will get returned with the JSON representation of

ML_DECIBEL_8844_PARENT.RFL.



Appendix D: API Documentation for RFL Factor Rest Service

1. Overview

This document describes the API Risk Engine will utilise to access published RFL factors that have been stored in Sandra.

2. HTTP Calls

The REST Service lives on the Quartz grid and can be found using Quartz's Service Discovery protocol[1]. The Service Discovery Protocol will provide the server name and port which can be used to access the REST Service. Requests may be submitted to the service discovery server and will be automatically redirected to the volatility service.

- Requests to the REST Service are made using HTTP.
- No HTTP headers in the request are needed.
- No message body data is needed.
- Returns from requests are in JSON format.

The following protocols for requesting RFL factors will be implemented

- **GET / CreditRFLFactors /{DataSet}/{MarketDate} HTTP/1.1**
    - Returns an array containing all credit RFL factor objects for dataset {DataSet} and market date {MarketDate}
- **GET /CreditRFLFactors/{DataSet}/{MarketDate}/{Name} HTTP/1.1**
    - Returns the credit RFL Factor object for the specified {DataSet} , {MarketDate}, and {Name}.

1 http://lnyce23217.bankofamerica.com:8181/docs/sphinx/html/components/servicediscovery.html

{MarketDate} must be in YYYYMMDD format, e.g. 20121130.

{DataSet} is the market dataset in Sandra, e.g. CREDIT_NY

{Name} is the top-level identifier of the credit RFL factor, e.g.

ML_DECIBEL_8654_PARENT

* Note, the rest service will auto detect requests from desktop browsers (e.g. Firefox, Internet Explorer) and render responses in HTML for viewing convenience. Non-rendered, native JSON messages can be forced by appending fmt=json as a request parameter: GET /CreditRFLFactors/CREDIT_NY/20121205**?fmt=json**

GET /creditRFLFactors/CREDIT_NY/20121130/ML_DECIBEL_8654_PARENT HTTP/1.1

{
'Name': 'ML_DECIBEL_8654_PARENT',
'Entry': '2012-11-30',
'DataSet': 'CREDIT_NY',
'QuotedBy': 'luyang.zhang',
'QuotedAt': '2012-11-30T16:51:14Z',
'ScaleBase': ['4.454431266528465', '0.13498698341468446',
'0.1366913808148539'],
'ScalesShadow': ['4.454509804485647', '0.1345869487137669',
'0.13635125877537008'],
'ThresholdsBase': ['-4.0', '2.644373074971677E-4'],
'ThresholdsShadow': ['-4.0', '4.254511530306401E-4'],
'CalibrationParams': {}

Example 1 HTTP Request and Response for a single credit RFL factor.

GET /creditRFLFactors/CREDIT_NY/20121130 HTTP/1.1

{
'Name': 'ML_DECIBEL_8654_PARENT',
'Entry': '2012-11-30',
'DataSet': 'CREDIT_NY',
'QuotedBy': 'luyang.zhang',
'QuotedAt': '2012-11-30T16:51:14Z',
'ScaleBase': ['4.454431266528465', '0.13498698341468446',
'0.1366913808148539'],
'ScalesShadow': ['4.454509804485647', '0.1345869487137669',

'0.13635125877537008'],
'ThresholdsBase': ['-4.0', '2.644373074971677E-4'],
'ThresholdsShadow': ['-4.0', '4.254511530306401E-4'],
'CalibrationParams': {}
}

{
'Name': 'ML_DECIBEL_8654_PARENT',
'Entry': '2012-11-30',
'DataSet': 'CREDIT_NY',
'DuotedBy': 'luyang.zhang',
'QuotedAt': '2012-11-30T16:51:14Z',
'ScaleBase': ['4.454431266528465', '0.13498698341468446',
'0.1366913808148539'],
'ScalesShadow': ['4.454509804485647', '0.1345869487137669',
'0.13635125877537008'],
'ThresholdsBase': ['-4.0', '2.644373074971677E-4'],
'ThresholdsShadow': ['-4.0', '4.254511530306401E-4'],
'CalibrationParams': {}
}

Example 2 HTTP Request and Response (truncated) for all credit RFL factors for dataset and market date

3. Quartz Discovery Service / Redirection

All Credit RFL factor REST services will be registered with the Qz service directory at http://qzsd.bankofamerica.com:8814/. Service clients can access all rest services through the discovery service URL. Requests will be automatically redirected to the corresponding service under the registered service name.

qzsd.bankofamerica.com:8814/**sct-rfl-rest-dev**/creditRFLFactors/CREDIT_NY/20121130/ML_DECIBEL_8654_PARENT

The request url above will automatically be redirected to the current grid server and port corresponding to the service name, **sct-rfl-rest-dev**:

lnyce23220.usnycbt.amrs.bankofamerica.com:10008/creditRFLFactors/CREDIT_NY/20121130/ML_DECIBEL_8654_PARENT

Example 3 HTTP Request and Response for browser listing available databases

To distinguish environments, credit RFL factor rest services will follow a naming convention.

Environment	**Service Name**
Dev	sct-rfl-rest-dev
QA	sct-rfl-rest-qa
Production	sct-rfl-rest-prod
<env>	sct-rfl-rest-<env>

Table 1 REST service environment naming convention.