DealCapture By OnBelay

Michael Le Feuvre
May, 2024

The DealCapture by OnBelay project implements a back-end solution starting with the capture of Physical Commodity Deals (Natural Gas and Electricity) right through to the generation and valuation of daily positions.

DealCapture is not meant to be a complete solution but it does provide an excellent starting point for developing an Energy Trading Risk Management (ETRM) app for natural gas, electricity, etc.

DealCapture consists of the midtier and database design and implementation scripts but does not include a user interface although it does provide a RESTful API using JSON to support most modern UI Frameworks nor does DealCapture implement any scripts to calculate Value-at-Risk, Financial Positions, etc. but it does capture the unique risk factors required for such analytics.

DealCapture is an open source project available on github at
https://github.com/OnBelayConsulting/dealcapture

HISTORY

The DealCapture project is an open-source project available in GitHub and was developed from intellectual property developed by myself at OnBelay Consulting. I spent thirteen years as the Technical Lead and Senior Software Developer at SAS in the Calgary R & D office developing and supporting the ETRM SAS Commodity Risk product. Prior to working at SAS I worked on various other projects implementing digital wallets, equities trading engines, pipeline contracts, etc.

The DealCapture project shares no direct DNA with the SAS Commodity Risk product, but it does incorporate more than twenty years plus of experience in the financial industry. The DealCapture approaches the ETRM requirements in fundamentally different ways from Commodity Risk as described below.

DealCapture captures only physical deals and does not include agreements. DealCapture includes the ability the specify costs, prices and quantities down to the daily and hourly level.

DealCapture has a different pricing model based on indices and price curves only. Commodity Risk included an intermediate entity, called curve definition that managed frequency.

DealCapture generates daily positions for gas and electricity with additional hourly position details if required. (This approach greatly reduces the number of positions for power.)

Price and Delivery Profiles are handled in DealCapture using a PowerProfile. A PowerProfile is directly associated to the various indices for different flows including off-peak, on-peak, settled, end-of-month, etc. DealCapture does not use commodity subtypes.

Hourly indices may be specified on any deal regardless of commodity.

WHATS IN THE PROJECT

The DealCapture project consists of the following modules:

1. Java Spring Boot application designed to work with OAuth2,
2. Core library (obcore on GitHub) that provides generic support for rich domain modeling and querying.
3. Database design and Liquibase database scripts that support multiple database creation and maintenance.
4. Gradle scripts for building and GitHub action scripts for running tests and deploying.

LIMITATIONS

The DealCapture project does not include a user interface. The DealCapture backend app does support REST style API using JSON to support easy integration into frameworks written in React, etc.

Design and Features

The DealCapture project is designed to support large scale ETRM requirements and integration into upstream applications using RESTful services, support for analytics such as MtM and settlement, advanced querying and efficient paging.

ANALYTICS

The DealCapture app calculates Mark to Market and Settlement positions for Physical Deals. (Other deal types may be added quite easily.)

Position generation creates price and foreign exchange (F/X) risk factors as a by-product and these risk factors combined with the associated positions may be used to calculate Value-At-Risk (VaR), Financial Positions, and other analytics in any language or tool as desired.

Position generation and valuation is heavily optimized to be highly performant. For example, ten million daily positions are generated and valued (MtM and Settlement) in under twenty minutes using a single standard server.

ENERGY COMMODITIES

DealCapture supports energy commodities such as Natural Gas and Electricity.

Support for all commodities includes support for complex price modeling using basis relationships, benchmark relationships and price discovery.

Price discovery, the search for prices from additional sources when prices are not initially available is supported through two mechanisms.

The first mechanism uses price escalation in the associated price curve. If an hourly price is not found then the application looks for a daily price and then a monthly price. Prices are stored with frequency in the same price curve.

The second mechanism uses the benchmark relationship if present between price indices. The search for prices will continue through the relationships until a price is found or there no other relationships available.

Fixed prices, costs and quantities may be specified down to the daily and hourly level. This feature is implemented using a spreadsheet like approach with day and hourly slots for set-up.

Electricity based deals are supported using a Power Profile. A Power Profile identifies what price types (called PowerFlowCodes) are appropriate for each hour slot, when power flows and what price indices are the source of prices for each Power Profile. Associated price indices may be hourly, daily or monthly and do not require a specific commodity subtype.

Positions and risk factors for Power profiles are generated separately and in advance of deal positions. This feature supports separate analysis of Power Profiles and greatly reduces the computing time of deal positions.

POSITIONS

DealCapture generates daily positions as the default. If the deal is associated to a PowerProfile or an hourly index or supports separate hourly fixed values then additional hourly position details by day records are generated. (This approach will generate one record versus twenty-four records which greatly reduces the processing time.)

Position generation will also create price and F/X risk factors if necessary. Price and F/X risk factors are unique and are likely associated to more than one deal position.

Positions are used for both MtM and settlement.

MARK-TO-MARKET

The process to calculate Mark-to-Market (MtM) for physical deals is as follows:

1. Generate Power Profile positions if power Profiles are in use.
2. Generate Deal Positions
3. Load price and F/X curves with latest values
4. Value price and F/X risk factors. (Assign a price based on price discovery)
5. Value Power Profile positions
6. Value Deal Positions.

All the above steps may be executed with a single command or may be executed individually. For example, risk factors and positions may be re-valued at any time after the initial generation.

SETTLEMENT

Total cost, commodity settlement and total settlement amounts are calculated for any positions generated in the settlement currency specified on the deal. Note that positions may be generated in any configured currency and in multiple currencies but usually only one set of positions are used for settlement downstream of the deal capture app.

APIS

DealCapture is designed with the following layers:

1. Controllers – define REST endpoints protected by OAuth2.
2. RestAdapters – interface between Controllers and Services.
3. Snapshots – POJO classes that map directly to JSON payloads used in APIs.
4. Services – defines various services such as DealService, PositionService, etc.
5. Repositories – logical abstraction over the ORM to access database functions.
6. Domain Classes – Rich domain objects that encapsulate complex business object.

DealCapture follow the REST style of APIs with a few exceptions that better support managing a complex data graph. The APIs are callable from an client that can obtain an OAuth2 token and are easily test with API test applications such as Postman.

SECURITY

DealCapture is designed to be secured with OAuth2. Any OAuth2 implementation should work out of the box. OAuth2 has become the default standard in Cloud Native and on-premise applications as it supports security across cooperating applications and integrates easily with security products such as Microsoft Active Directory.

CLOUD NATIVE

DealCapture is completely Cloud Native and will run in a variety of deployment configurations including in Microservices Kubernetes environments or it will run on a stand-alone server.

The term Cloud Native is not well defined but in the context of DealCapture it means that the application will run in a cloud computing environment. For example, DealCapture is designed to run in Kubernetes. The DealCapture database is implemented in Liquibase scripts that may be executed in a sidecar as part of application start-up. DealCapture will also run with no modification on a virtual machine or stand-alone server as well.

DealCapture is built using the Spring Boot Framework available from SpringFramework. Spring Boot based apps are easily containerized generally using build tools such as Gradle. The Spring Framework project is the major open source project used by Java applications world-wide.

The DealCapture project consists of two applications:

- DealCapture app,
- Organizations app

Organizations such as Companies and Counterparties are maintained in a separate application or Microservice. Any additions or changes to organizations in the Organizations app are automatically synchronized to the DealCapture app using RabbitMQ. This is not a necessary part of the architecture and the Organization app may be combined with the DealCapture app quite easily.

The DealCapture project takes this approach to demonstrate how to build a Microservice ecosystem using RabbitMQ as the messaging backbone and the organization module may be rolled into DealCapture with minimal effort. (Why you may or may not want to do this is subject to a separate discussion of course.)

ADVANCED QUERYING

The advanced query engine supports SQL-like queries (without the security issues) right through from the REST services to the domain service layer. This approach is very flexible and it supports paging to make it very performant.

For example this query defined on the fetch deals API will fetch all deals with start dates greater than Jan 1st, 2024, for counterparty TD with a ticketNo containing the letters “GH”.

```
WHERE startDate gt ‘2024-01-01’
  AND counterpartyName eq ‘TD’
  AND ticketNo contains ‘GH’
```

IMPORT/EXPORT

The current API supports importing basic physical deals via the comma-separated-value (CSV) file format easily generated from a Spreadsheet application. The format may be extended to support important various deals as required.

The current API also support exporting deal positions in CSV file format as well.

DATABASE

The Java database creation and maintenance are implemented using Liquibase. Liquibase maintains a record of all database changes and supports bundling both programming and database updates in one source check-in for both ease of deployment and roll-back.

The Liquibase scripts are executed within a Gradle script to simplify the execution and deployment and include best practices for managing updates.

FAQ

I need to support multiple DBMS such as SQL Server, Oracle, etc.

The DealCapture project is based on the Spring Framework which incorporates the Hibernate ORM and uses the Liquibase toolset to support a database abstraction layer. DealCapture supports PostgreSQL and SQL Server in the default implementation and with minimal configuration will support other DBMS such as Oracle.

My project requires support for other deal types such as Financial Swaps, Options, Transportation, etc.

DealCapture implements a complete backend solution for Physical Deals. Other Deals may be supported by extending from the generic Base Deal and then using the Physical Deal as a example.

Options will require additional programming to support option valuation. This step can be implemented in Java or can be implemented in an analytics specific language such as SAS or R.

Why is DealCapture security designed around OAuth2?

OAuth2 is the *de facto* standard for most cloud native applications and I strongly urge you to consider adopting this standard for your applications. There are both commercial and open source projects available that implement this standard and most projects support integration with Microsoft Active Directory, the other de facto standard for securing network objects.

What do you mean with the statement that DealCapture is a Cloud Native app?

The term Cloud Native generally means that the application will run on natively in most cloud environments including Kubernetes. That doesn't mean that DealCapture (or any application

based on Spring Boot for that matter) will not run as a stand-alone application on a physical or virtual machine.

The Gradle and GitHub action scripts included with the DealCapture project provide a good starting point to build a DevOps pipeline starting with a successful source code check-in to the automatic deployment into a Kubernetes installation.

My application needs to generate and value millions of positions in a short timespan.

Database inserts and updates for large volumes of data must be highly optimized an ORMs such as Hibernate are not designed to support large batch operations (there is support for batching in Hibernate but it is still significantly slower.)

The DealCapture project demonstrates how to use the JDBC batch support to batch up inserts and updates using the JDBC API to support for large data operations. Overall performance is based on many things including the selected DBMS and the network but in my testing the JDBC batch approach is up to ten times faster than a naïve approach using the out of the box Spring Framework/Hibernate.

How do I license DealCapture for my project and is there any support going forward?

The DealCapture project is open source project licensed under the Apache 2.0 License. Clone the DealCapture project to make a version that will be yours. I will be available for support (free or paid depending on the ask) for some things but since you will have a complete copy of the source code your copy is your responsibility to maintain.

Does DealCapture support calculating VaR and financial positions?

DealCapture does not implement scripts to calculate Value at Risk (VaR) or Financial Positions BUT the generate positions step identifies and creates a unique set of price and foreign exchange risk factors that are associated to the generated positions.

The combination of positions and risk factors provides the starting point for all subsequent analytics such as VaR, CVaR, etc. These analytics are typically implemented in languages such as SAS or R.

Are you available for short-term or long term consulting engagements?

The short answer is yes for short-term engagements and a maybe for longer term based on the duration and location.