Quaternion™
RISK MANAGEMENT

Sensitivity Analysis, ISDA SIMM Benchmarking and Backtesting with RESTORE

ORE User Meeting – Frankfurt – 23 November 2018

Quaternion™
RISK MANAGEMENT

Agenda

- ISDA SIMM & AcadiaSoft
- Sensitivity Service
- Benchmarking and Backtesting Service
- Trade representation (ORE XML, FpML)
- ORE in a web services environment
- RESTORE
- Challenges with ORE

Quaternion™
RISK MANAGEMENT

ISDA SIMM

- Standard Initial Margin Methodology defined by ISDA
- Defines Initial Margin amounts to be posted between counterparties based on netting set sensitivities (delta, vega, etc)
- Can therefore be applied to any OTC derivatives portfolio
- Generally considered to be overly conservative, with some exception cases
- To Calculate SIMM, one must be able to calculate specific sensitivities, in particular sensitivities to par instruments (2Y Swap Rate not 2Y Zero rate)

Quaternion™
RISK MANAGEMENT

AcadiaSoft

AcadiaSoft, Inc. is uniquely focused on delivering margin automation and standards for counterparties engaged in collateral management.

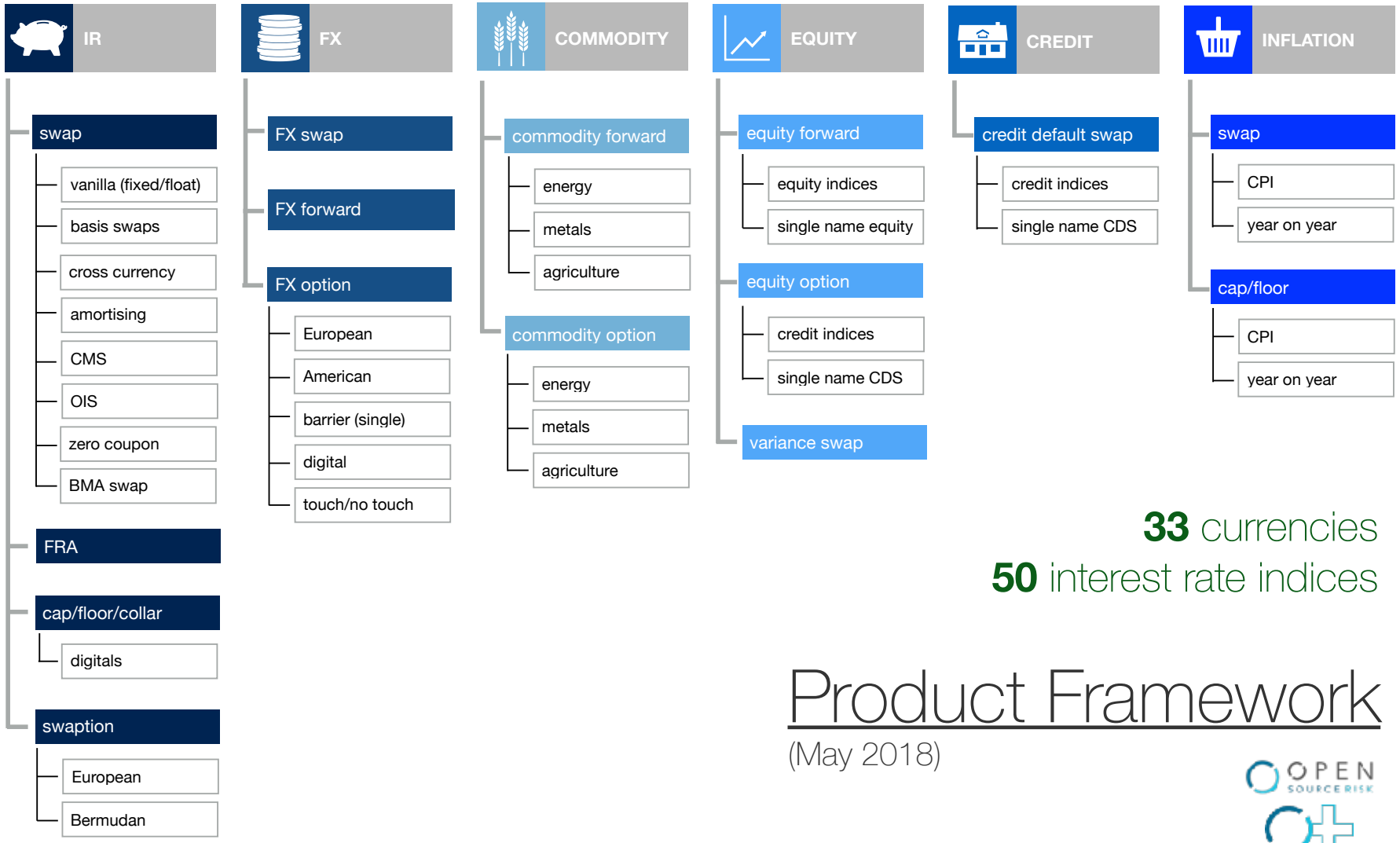
AcadiaSoft Hub is the industry's only straight through margin processing solution. Developed in collaboration with some of the industry's leading financial institutions, AcadiaSoft Hub is a one-stop solution for meeting the increased margin workload required for compliance with upcoming rules on margining non-cleared derivatives

Quaternion™
RISK MANAGEMENT

Sensitivity Service

- Service to calculate the SIMM required sensitivities and then the IM for a given portfolio
- Daily service, client upload portfolio details overnight and IM is calculated
- Went live in September 2018
- Built by Quaternion and AcadiaSoft using ORE

- Hosted on AcadiaSoft's platform, which is microservices based
- Market data sourced primarily from Reuters
- Clients can upload portfolio in ORE XML format or FpML



33 currencies

50 interest rate indices

Product Framework

(May 2018)

Quaternion™
RISK MANAGEMENT

Sensitivity Service – Process outline

- Outline of daily process:
  - Trades are stored in ORE XML format in a database
  - Trades loaded in ORE
  - Configuration files are dynamically built (based on portfolio) and passed into ORE
  - NPV and raw sensitivities are calculated by ORE
  - Par sensitivities calculated by transforming raw values
  - CRIF file generated and posted upstream in AcadiaSoft
  - SIMM calculated by AcadiaSoft

- The transparency of the underlying pricing models and methodology can be of great benefit to clients

Quaternion™
RISK MANAGEMENT

Benchmarking and Backtesting Service

- Service to preform a historical backtest of SIMM for every netting set in a portfolio. i.e. compare todays SIMM with historical PnL moves for that netting set
- Quarterly service, client upload portfolio details overnight and report is calculated
- Went live in September 2018
- Built by Quaternion and AcadiaSoft using ORE
- Also compares SIMM to other IM calculations (Historical VaR variants, CCP)
- Aimed at satisfying regulatory requirements an institution might have

- Similar integration to Sensitivities

Quaternion™
RISK MANAGEMENT

Backtesting – Process outline

- Historical market data dating back to 2008 has been sourced
- Using ORE we have bootstrapped a full set of curves for each date (2,579)
  - ~80 IR curves – using basis swaps, xccy, etc
  - ~10 Cap and Swaption surfaces, ~30 FX Vol surfaces
  - ~300 Default Curves, ~300 Equity, ~5 Inflation Curves
  - This data is stored as an ORE Scenario in a database
  - XOIS with different base currencies (EUR, USD, GBP, etc) depending on CSA currency
- In total 58 different curve configurations were needed due to changing market data over the 10 years
- Here, the strict and rigid bootstrap framework in ORE was difficult to work with

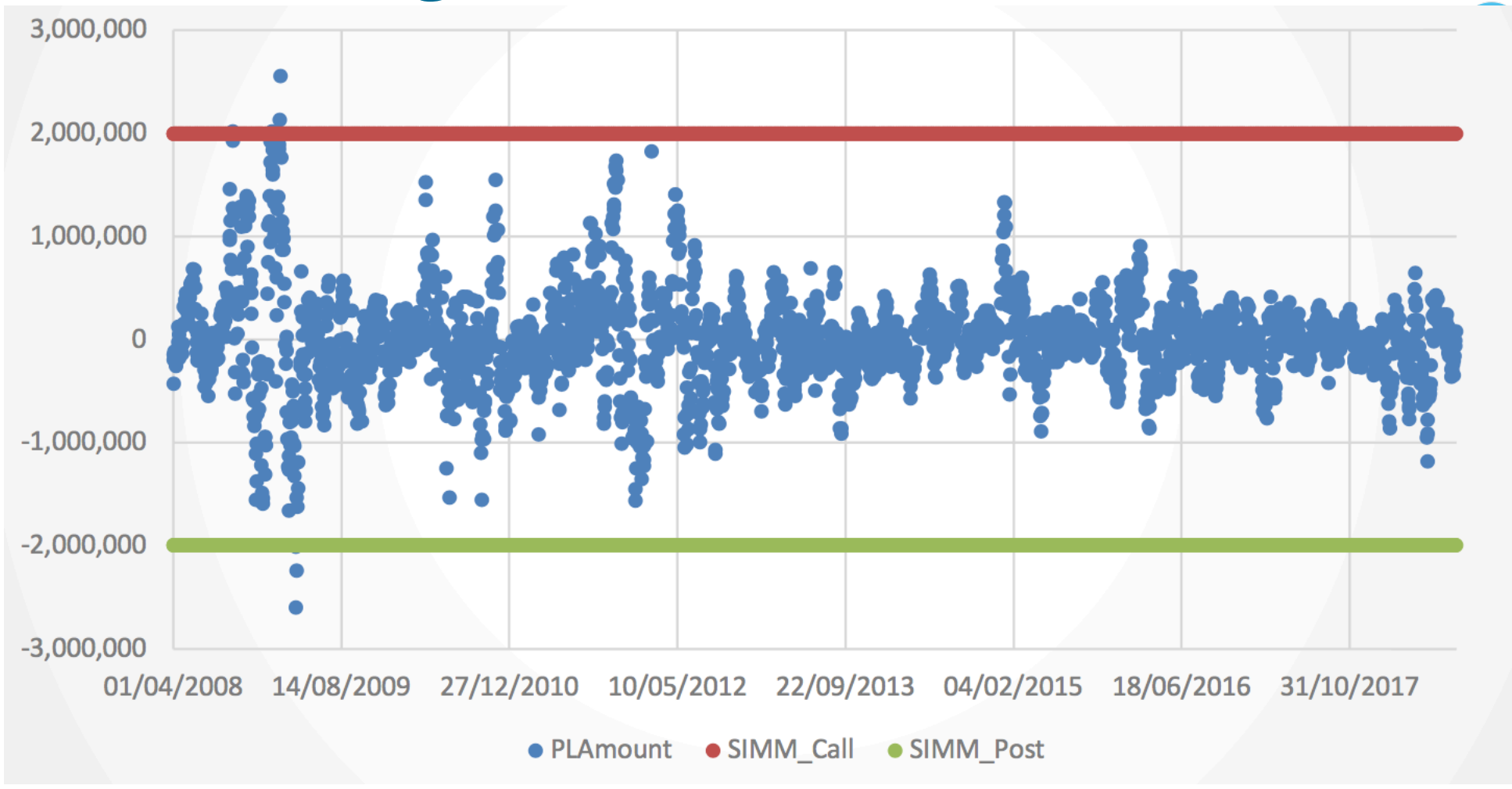
Quaternion™
RISK MANAGEMENT

Backtesting – Process outline

- Outline of quarterly process:
  - Trades are stored in ORE XML format in a database, loaded into ORE
  - Configuration files are dynamically built (based on portfolio) and passed into ORE
  - SIMM is calculated with ORE inputs
  - Historical scenarios are used to compute 10 days moves and these moves are applied to todays market (also an ORE Scenario)
  - Scenario Algebra is used to generate PnL moves today from historical market moves
  - ORE Scenario and Valuation framework was utilised to build a PnL vector
  - Backtesting is then the normal Basel Red, Amber and Green statistics
  - This is sometimes referred to as a Static Backtest
  - Automatic run is a few hours, a final report is compiled and delivered to the client.

Quaternion™
RISK MANAGEMENT

Backtesting – Historical PnL



Quaternion™
RISK MANAGEMENT

Backtesting – Process outline

- Outline of quarterly process:
  - Trades are stored in ORE XML format in a database, loaded into ORE
  - Configuration files are dynamically built (based on portfolio) and passed into ORE
  - SIMM is calculated with ORE inputs
  - Historical scenarios are used to compute 1 or 10 days moves and these moves are applied to todays data (also an ORE Scenario) to generate a new Scenario
  - Scenario Algebra is used to generate PnL moves today from historical market moves
  - ORE Scenario and Valuation framework was utilised to build a PnL vector
  - Backtesting is then the normal Basel Red, Amber and Green statistics
  - This is sometimes referred to as a Static Backtest
  - Automatic run is a few hours, a final report is compiled and delivered to the client.

Quaternion™
RISK MANAGEMENT

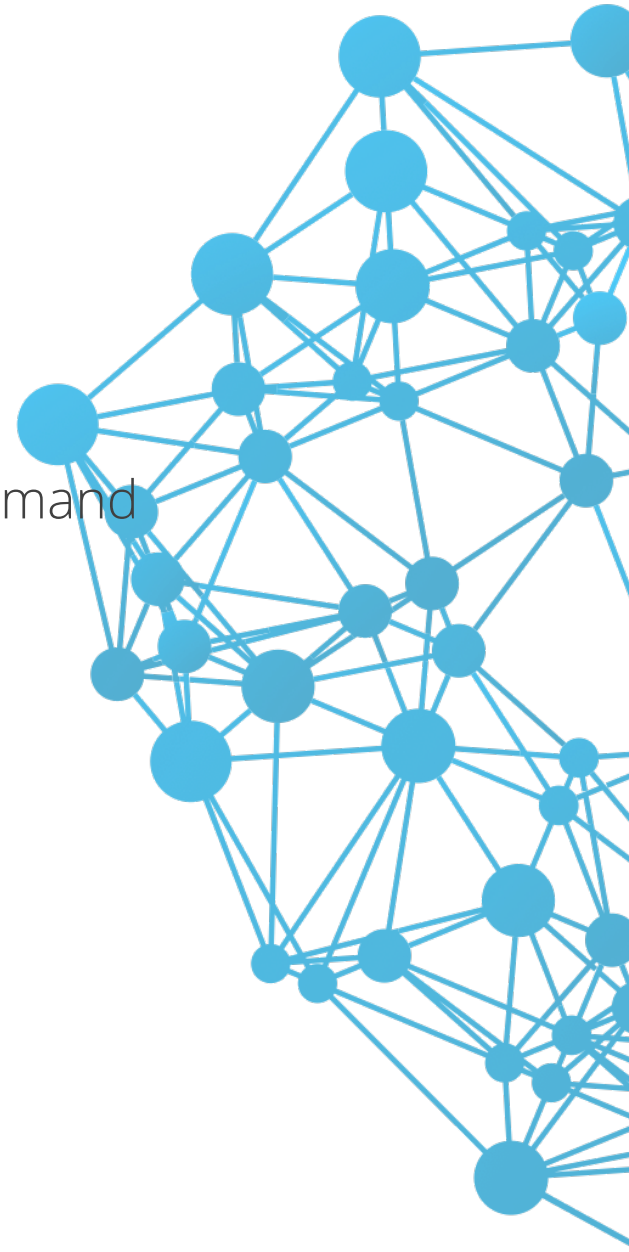
Trade representation (ORE XML, FpML)

- There is no standard trade representation format that everyone uses
- FpML is the closest to an open standard, however adoption of FpML is low and there are multiple versions with different compatibility issues.
- FIX is limited in scope

- Both Services with AcadiaSoft allow clients to upload an ORE portfolio xml file, and this choice is proving popular.
- In this scenario, clients are treating ORE XML as a **standalone trade representation** and not looking at ORE as a whole
- It is possible that ORE XML will grow in it's own right and sit beside FpML, independent of the actual library

Quaternion™
RISK MANAGEMENT

ORE in a web services environment



- ORE is a set of quantitative libraries with a single application, a simple command line application that takes no interactive user input and does not persist.
- The command line app is designed to showcase ORE and run examples, however the libraries themselves are capable of being used in other environments:
  - Desktop applications with a GUI
  - Distributed applications and servers
  - Web Services
  - Mobile phone apps
- Quaternion has developed a set of Web Services around ORE and other proprietary libraries

Quaternion™
RISK MANAGEMENT

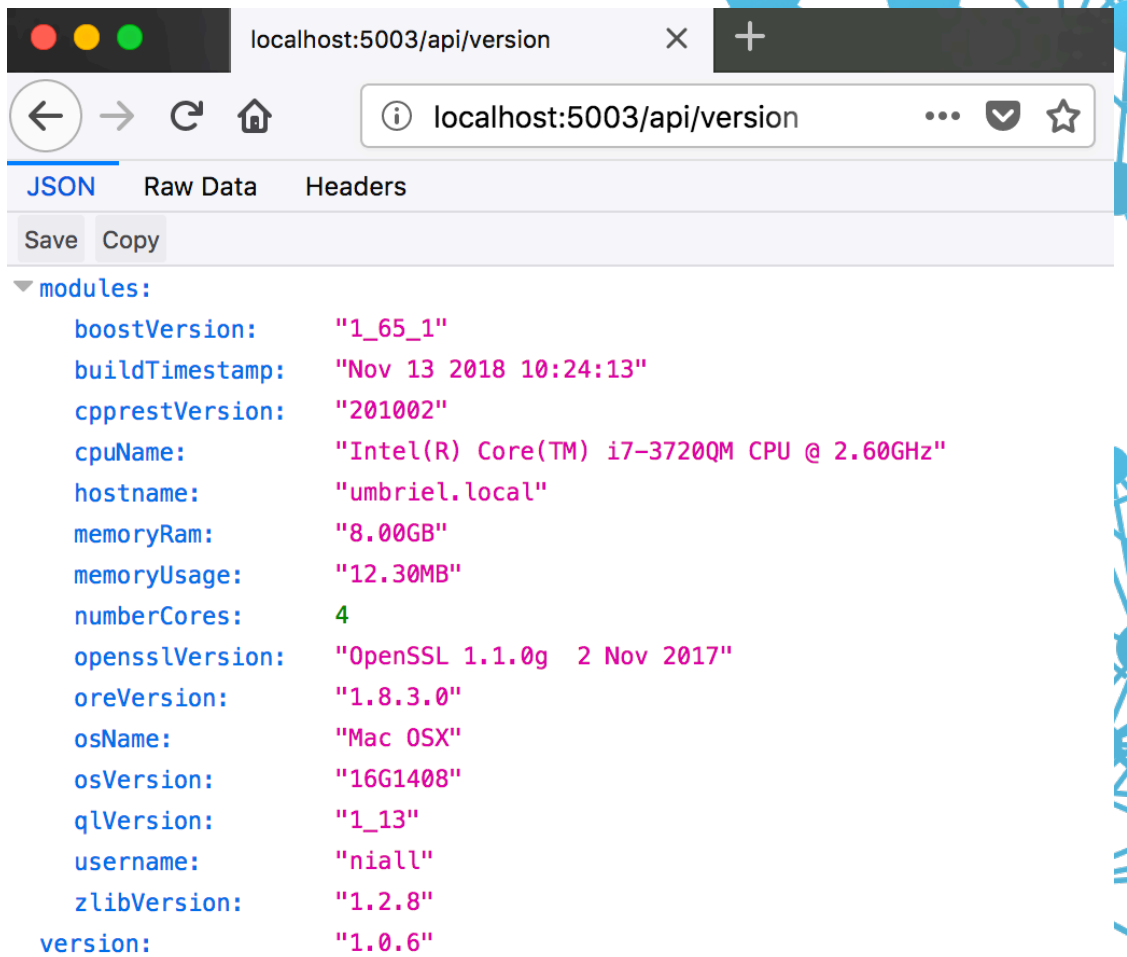
RESTORE

- Adding a RESTful API to ORE (”REST” + “ORE” = “RESTORE”)
- Developed a number of services (not quite microservices) for hosting data (market, trade, configuration), doing analytics (with ORE), hosting large cubes and persisting results
- Core service is the ORE pricer which does all analytics – this has been deployed in AcadiaSoft as part of this project
- Pricer is 100% C++, linking ORE with CppRestSDK (formerly Casablanca)
  - *The C++ REST SDK is a Microsoft project for cloud-based client-server communication in native code using a modern asynchronous C++ API design*
- Also uses zlib, OpenSSL (for SSL/TLS and JWT auth), and ORE+
- Other services have been developed with a combination of C# and Python

Quaternion™
RISK MANAGEMENT

RESTORE

- RESTORE pricer sits in a docker container, when spun up it launches a server which listens on a port for requests
- Is largely stateless and only responses to GET for version and some settings (e.g. log level)
- Possible to launch multiple pricers (either manually spinning up a few or using AWS elastic or k8s)
- CppRestSDK is a multithreaded framework, with a default pool of 40 listener threads, but ORE is not threadsafe so all ORE calls are locked



Quaternion™
RISK MANAGEMENT

RESTORE – sample POST for NPV & Sensi

POST | http://localhost:5003/api/pricer/clientid/analytics | Params | Send | Save

Authorization | Headers (1) | Body | Pre-request Script | Tests | Cookies | Code

form-data | x-www-form-urlencoded | raw | binary | JSON (application/json)

```
{
    "analytics" : "NPV,SENSITIVITY",
    "marketUri": "https://localhost:5001/api/marketdata/2018-09-28",
    "fixingsUri": "https://localhost:5001/api/fixings/2018-09-28",
    "curveConfigUri" : "https://localhost:5001/api/config/clientid/2017-12-29/todaysmarket",
    "conventionsUri" : "https://localhost:5001/api/config/clientid/2017-12-29/conventions",
    "pricingEngineUri" : "https://localhost:5001/api/config/clientid/2017-12-29/pricingengine",
    "asof": "2018-09-28",
    "baseCurrency": "USD",
    "portfolioUri" : "https://localhost:5001/api/trades/clientid/portfolioxml/2018-12-29?nettingSetId
        =testNS1"
}
```

Quaternion™
RISK MANAGEMENT

RESTORE – Sample Response


```
"npv": [
    {
        "baseCurrency": "USD",
        "counterpartyId": "CP_A",
        "jobId": 0,
        "maturity": "2027-01-04T00:00:00.000Z",
        "maturityTime": "8.268493151",
        "nettingSetId": "NS_A_1",
        "notional": "10000000.000000000",
        "notionalBase": "11610356.437942646",
        "npv": "-813610.887393852",
        "npvBase": "-944631.240443344",
        "npvCurrency": "EUR",
        "tradeId": "Trade_A1_1",
        "tradeType": "FxForward"
    },
    {
        "baseCurrency": "USD",
        "counterpartyId": "CP_A",
        "jobId": 0,
        "maturity": "2019-01-04T00:00:00.000Z",
        "maturityTime": "0.268493151",
        "nettingSetId": "NS_A_1",
        "notional": "10000000.000000000",
```

```
{
    "baseNpv": "186642.635088068",
    "currency": "USD",
    "delta": "-15.046803577",
    "factor_1": "DiscountCurve/EUR/6/3Y",
    "factor_2": "",
    "gamma": "0.001213145",
    "isPar": "false",
    "jobId": 0,
    "shiftSize_1": "0.000100000",
    "shiftSize_2": "0.000000000",
    "tradeId": "Trade_A1_14"
},
{
    "baseNpv": "186642.635088068",
    "currency": "USD",
    "delta": "1866.426350881",
    "factor_1": "FXSpot/EURUSD/0/spot",
    "factor_2": "",
    "gamma": "0.000000000",
    "isPar": "false",
    "jobId": 0,
    "shiftSize_1": "0.011610356",
    "shiftSize_2": "0.000000000",
    "tradeId": "Trade_A1_14"
},
```


Quaternion™
RISK MANAGEMENT

RESTORE – Pricer integration with ORE

- We use CppRestSDK to send requests to other services for data
  - Market and Fixing data is returned in JSON or CSV – using a subclass of ORE Loader
  - Trade and Config data are returned as XML – loaded in ORE using `XMLSerializable::fromXMLString()`
- All components are loaded in memory – no files are loaded – and then ORE is invoked (under a lock)
- RESTful call blocks until everything is calculated
- ORE Report class framework was used to convert reports to JSON (using CppRestSDKs JSON library)
- Custom Logger added that wraps boost::log, Ideal for long lived servers, with automatic log rotation (no compression) and is thread safe.

Quaternion™
RISK MANAGEMENT

RESTORE – Distributed Computation

- At the core of exposure and large sensitivity runs, is the ORE valuation engine, which loops over scenarios and prices the portfolio.
- RESTORE allows us to distribute this “pricing under scenarios” step to a cluster of pricers, and gather the results back in a single cube
- Some overhead with the communication around sending scenarios out and collecting cube results
- Also possible to split up a portfolio and send subset to different nodes.

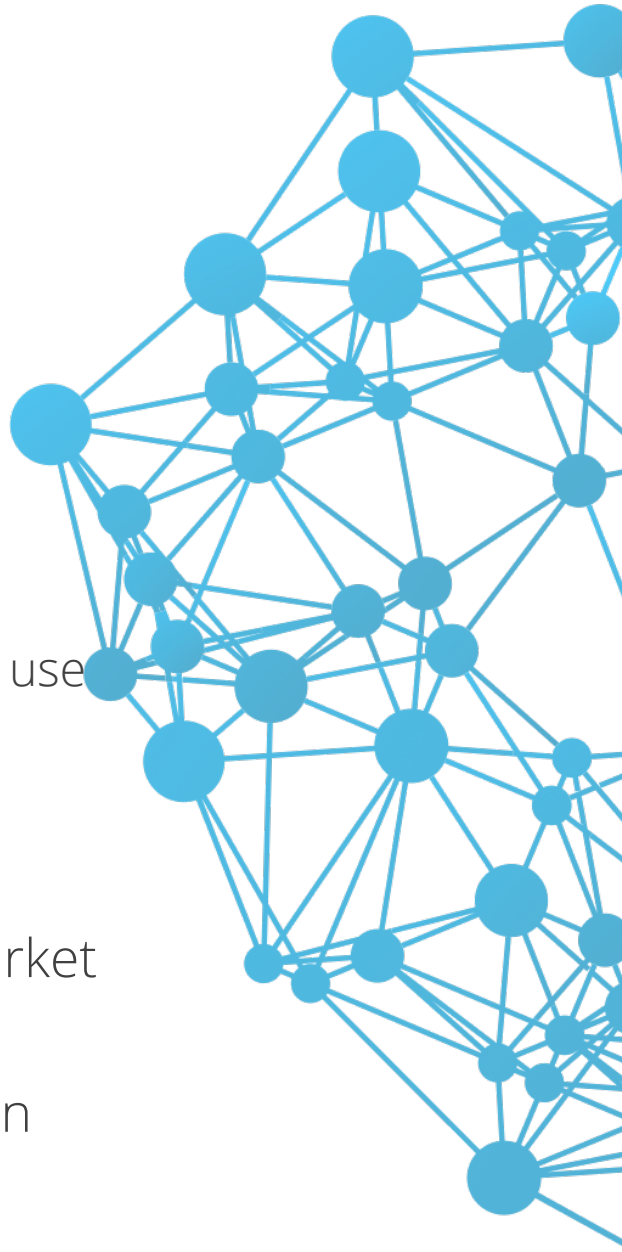
Quaternion™
RISK MANAGEMENT

Challenges with ORE

- Curve Bootstrap framework is strict
  - When one curve fails the whole process stops, hard to collect all errors in one go
  - With historic data not all points are always available, would be nice to make some points optional (coming in v4)
- Configuration files are powerful but complicated, if you do not know your portfolio or product scope in advance how can they be generated?
- An always on, dynamic service, needs informative logging – ORE log levels vary and do not get passed down into QuantLib or QuantExt (e.g. I want to log what implied volatility is used when pricing an option)
- Using ORE in a multi-threaded framework like CppRestSDK requires mutex locks
- Its not currently possible to ask a Portfolio what fixings it will require, you must load them all (**ideas welcome here!!**)

Quaternion™
RISK MANAGEMENT

Challenges with ORE



- Single discount curve per currency
  - Good for exposure and other use cases
  - For a portfolio of FX forwards, Options and XCCY Swaps, some people want to use different curves for each.
  - In some emerging markets mixing NDF with on-shore swaps requires different curves
- Cap and Swaption surfaces are only defined for a single tenor (the ORE market interface does not allow a separate 3M and 6M Cap)
- ORE is only in C++, which is limiting when looking to integrate into a modern framework, most RESTful services are built in Java, C# or python. While it is possible to do all in C++ you don't get as much for free (e.g. swagger)

Quaternion™
RISK MANAGEMENT

The end

- Questions?
- Thank you

Firm locations

Quaternion™ Risk Management is based in four locations :

Ireland

54 Fitzwilliam Square,
Dublin D02 X308,
Ireland.

+353 1 678 7922

United Kingdom

Martin House,
5 Martin Lane,
London EC4R 0DP

+44 2077121645

USA

24th floor,
World Financial Centre,
200 Vesey Street,
NY 10281-1004.

+1 646 952 8799

Germany

Maurenbrecherstrasse 16,
47803 Krefeld,
Germany.

+49 2151 9284 800

Quaternion ™ is a sponsor of **opensourcerisk.org** partnering with Columbia University