Reference Data Service

Installation and Developer’s Guide

Release 1.0

Monday, August 14, 2017

Table of Contents

[Overview 4](#_Toc490468765)

[1.1 Packaging 5](#_Toc490468766)

[1.2 System Runtime requirements 5](#_Toc490468767)

[1.2.1 Hardware 5](#_Toc490468768)

[1.2.2 Operating Systems 6](#_Toc490468769)

[1.2.3 Java Runtime Environment (JRE) 6](#_Toc490468770)

[1.3 Configure and run the services. 6](#_Toc490468771)

[1.3.1 Configure Service 6](#_Toc490468772)

[1.3.2 Running the service 9](#_Toc490468773)

[1.3.3 Service Logging 9](#_Toc490468774)

[1.4 Source code 10](#_Toc490468775)

[2.1 Audit Service Reference implementation 10](#_Toc490468776)

[Audit Store API 10](#_Toc490468777)

[Audit Query API 10](#_Toc490468778)

[2.2 Party Service Reference Implementation 12](#_Toc490468779)

[The Party Service provides the following rest API for consumption by the WebTool. 12](#_Toc490468780)

[Get Party API 12](#_Toc490468781)

[Party Query API 12](#_Toc490468782)

[2.3 Operational Data Service Reference implementation 13](#_Toc490468783)

[FX Rate Quotes API 13](#_Toc490468784)

[Participants API 13](#_Toc490468785)

[Test User Credentials API 14](#_Toc490468786)

[Appendix 15](#_Toc490468787)

[A.1 General Architecture 15](#_Toc490468788)

[A.1 Audit Service interfaces and supporting classes 16](#_Toc490468789)

[A.2.1 Audit Service Store Override SPI implementation 16](#_Toc490468790)

[A.2.1 Audit Service Query Override SPI implementation 21](#_Toc490468791)

[A.3 Sample Reference Data Service yaml configuration file 24](#_Toc490468792)

[A.4 Reference Data Services Sample log4j2 file 25](#_Toc490468793)

[A.5 Audit Reference Service 25](#_Toc490468794)

[Store Audit Record API 25](#_Toc490468795)

[Query Audit Records API. 27](#_Toc490468796)

[A.8 Party Service 27](#_Toc490468797)

[Get Single Party Rest API 27](#_Toc490468798)

[Query Parties Rest API 28](#_Toc490468799)

[A.6 Operation Data Reference Service 30](#_Toc490468800)

[Participants API 30](#_Toc490468801)

[FX Rates Quotes API 31](#_Toc490468802)

[Boot Strap Test User Credentials API 32](#_Toc490468803)

# Overview

The primary purpose of the document is to describe the installation and integration of the Reference Data service implementations which are essential functional operation of the Adept WebTool UI.

A Client is free to implement the required APIs using any technology stack they deem appropriate with the one caveat that those service endpoints must support the Webtool UI’s functionality providing both the exact REST format JSON request and response structures.

Each reference service implementation is a Java 8 compatible Maven 3.x project and ships ready to run or can be recompiled using the favorite IDE of choice of the Client’s development staff.

In the proceeding sections of this document each service will be discussed in turn with breakouts of every api ( input parms , json request format and reposne format.

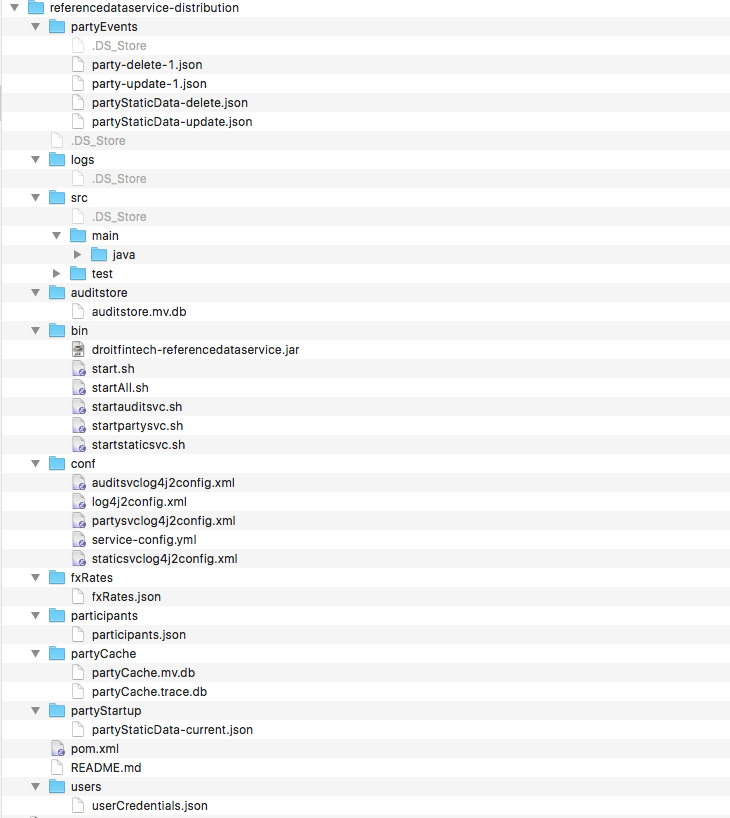
The current set of reference services covered in this document are as follows:

* Party Query Services
* Operational Data Service ( Participants , Users , FX quotes )
* Audit Decision Persistence and Query service
* Defaulting Service

## 1.1 Packaging

The Reference Data Service distribution is shipped as a bundle of binary jars, Source code and JSON samples all wrapped up in a tar gzip file. The default installation would be to simply untar the distribution tar.gz file and run everything in place.

The distribution jar has the following layout;



## 1.2 System Runtime requirements

### 1.2.1 Hardware

The following are the minimal hardware requirements for a Production environment:

* Cores - minimum of 4.
* RAM - Minimum of 4g
* 500 MB disk space.

### 1.2.2 Operating Systems

The Reference Data Services are operating systems agnostic and can run anywhere the JVM is installed.

The following list of operation systems are all suitable for runtime operation;

* RedHat Linux ( REHL 7+) ( tested )
* Suse Linux 12+
* Ubuntu 16+ ( Tested defsult )

### 1.2.3 Java Runtime Environment (JRE)

Oracle or Oracle-compatible Java v1.8.x or higher and maven 3.3.x and above.

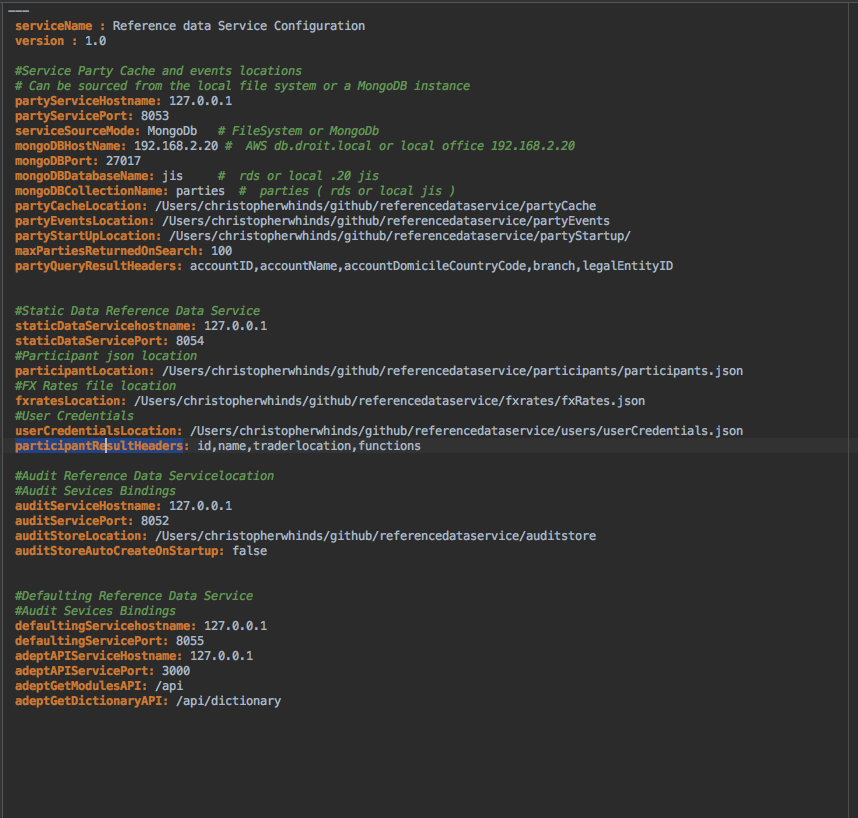
## 1.3 Configure and run the services.

### 1.3.1 Configure Service

Out of the box all the of the reference service startup bound to localhost and unless there is a conflict on listener port, should run without any intervention. A sample of the configuration properties file is located in the */conf* sub-folder of the distribution ( service-conf.yml ) .

If configuration tweaks are required please refer to section A4 of the appendix for the exact structure of the yaml file using the following table as a guide for setting each property.

Service configuration:



|  |  |  |  |
| --- | --- | --- | --- |
| Configuration Property Name | Required | Allowable Values | Comments |
| partyServiceHostname | Required | String | Hostname or |
| partyServicePort | Required | String | Port for the service to listen 8053 default |
| serviceSourceMode | Required | String, Filesystem or MongoDb | Source mode of the service to fetch party json. data. File System is defaualt |
| mongoDBHostName | Optional | String | Hostname of the MongoDB server |
| mongoDBPort | Optional | String | MongoDB server Port |
| mongoDBDatabaseName | Optional | String | MongoDB Database name. |
| mongoDBCollectionName | Optional | String | MongoDB data collection name |
| partyCacheLocation | Required | String | Full path to the location where the Party Cache will be stored. |
| partyEventsLocation | Required | String | Full path to the location where the Party Events will are stored. |
| partyStartUpLocation | Required | String | Full path to the location where the Party Startup JSON file will be stored. |
| maxPartiesReturnedOnSearch | Optional | String | Specifies maximum number of parties that can be returned on a search. Default is 100 |
| staticDataServicehostname | Required | String | Hostname or ip address that the service will bind to. |
| staticDataServicePort |  |  | Port for the service to listen 8054 default |
| participantLocation | Required | String | Full path to the location where the Participant Startup JSON file will be stored. |
| fxratesLocation | Required | String | Full path to the location where the FX Rates Quotes startup JSON file will be stored. |
| userCredentialsLocation | Required | String | Full path to the location where the development and testing User Credentials Startup JSON file will be stored. |
| auditServiceHostname | Required | String | Hostname or ip address that the service will bind to. |
| auditServicePort | Required | String | Port for the service to listen 8052 default |
| auditStoreLocation | Required | String | Full path to the location where the Audit Decisions Cache will be stored. |
| auditStoreAutoCreateOnStartup | Optional | Boolean,  true,false | Specifies that the Audit cache will be saved during server restarts. Default (false) |
| defaultingServiceHostname | Required |  | Hostname or ip address that the service will bind to. |
| defaultingServicePort | Required | String | Port for the service to listen 8055 default |
| adeptAPIServiceHostname | Required | String | Hostname the Adept API is listening on |
| adeptAPIServicePort | Required | String | Port the Adept API endpoint is listening on |
| adeptGetDictionaryAPI | Required | String | Adept API Dictionary Url |

### 1.3.2 Running the service

The following shell scripts are provided to run a properly configured service;

* **startpartysvc.sh** – starts the service has optional parm of “DEBUG” which places JVM into debug mode.
* **partysvc-shutdown.sh** – auto generated shutdown script.
* startstaticsvc.sh – starts the service has optional parm of “DEBUG” which places JVM into debug mode.
* **startstaticsvc.sh** – starts the service has optional parm of “DEBUG” which places JVM into debug mode.
* **staticsvc-shutdown.sh** - auto generated shutdown script
* **startauditsvc.sh** - starts the service has optional parm of “DEBUG” which places JVM into debug mode.
* **auditsvc-shutdown.sh** - auto generated shutdown script.
* **startdefaultingsvc.sh** - starts the service has optional parm of “DEBUG” which places JVM into debug mode.
* **defaultingsvc-shutdown.sh** - auto generated shutdown script

### 1.3.3 Service Logging

All Reference Service all supports logging through the Log4j2 logging API . See Appendix A6 for a sample of the configuration file. The API supports the standard log levels;

* INFO
* DEBUG
* TRACE

## 1.4 Source code

The ***Source*** folder within the distribution folder contains the actual java source used to build the reference services jar. This folder is a standard maven structured build format. Using the source plus the pom.xml provided at the root folder of the distribution a developer should be able to rebuild the runtime jar. To compile and build the source simply run **“ mvn clean install –DskipTests “ f**rom the root folder of the distribution to build the jar.

## 2.1 Audit Service Reference implementation

The Audit Service Reference implementation is a Java 8 Embedded Jetty HTTP/Jersey based rest API. The API will receive and respond with structured content in a JSON standard format. See Appendix A7 for samples. The following table is a description of the APIs supported in the reference implementation of this service. The Audit Service always stores and retieives complete full adept decision. There is now functionality that provides partial shot versions of decisions.

### Audit Store API

**Protocol** : HTTP and HTTPS

**Method**: POST

**URL** : http://[hostname]:[port]/audit/store

Request Headers: Content-type : application/json

Accept: application/json

BODY : should contain the JSON structure see Appendix A7 for samples.

**Expected HTTP status response code**

**200** -> OK : indicates that the audit records was stored successfully

**500** -> Server Internal error: indicating that store function failed, check the logs for errors

### Audit Query API

**Protocol** : HTTP and HTTPS

**Method**: GET

**URL** : http://[hostname]:[port]/audit/query/[--- & delimted list of query parms as name=value pairs]

Query Parms:

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type / Format | Sample and Comments |
| decisionId | String | String GUID of the decision |
| decisionDateStart | String – “2017-01-01” | IOS Standard format |
| decisionDateEnd | String – “2017-01-01” | IOS Standard format |
| submissionDateStart | String – “2017-01-01” | IOS Standard format |
| submissionDateEnd | String – “2017-01-01” | IOS Standard format |
| externalTradeID | String – “123456ABC” | Trade id from the source system |
| groupID | String – “123456ABC” | Group ID from the source system. |
| assetClass | String – “Equity” | Droit Asset Class specification |
| product | String – “IRSwap” | Droit Product specification |
| subProduct | String – “FIxedFLoat” | Droit Sub-Product specification |
| contraPartyID | String | A ContraParty ID |
| counterPartyID | String | A CounterParty ID |
| trader | String | Participant Trade Name |
| salesPerson | String | Participant Sale Person Name |
| userRole | String | Droit Specified Role |
| userID | String | Droit User ID |
| overridden | String | True or False. |
| allowedToTrade | String | True or False, If the trade was allowed. |
|  |  |  |

Request Headers: Content-type : application/json

Accept: application/json

BODY : should contain the JSON structure see Appendix A7 for samples.

**Expected HTTP status response code**

**200** -> OK : indicates that the audit records was stored successfully

**404** -> NOT FOUND : Search operation was not successfully , No matching records found

**500** -> Server Internal error: indicating that store function failed , check the logs for errors

## 2.2 Party Service Reference Implementation

The Party Service rerference implementation is a Java 8 Embedded Jetty HTTP/Jersey based rest API.

The primary function of service is to supplies the webtool with query-able Contra/Counter Party Data as data attributes in form of structured JSON data. When a user searches for parties from the GUI the WebTool queries the Party Service through REST APIs with the returned results selectable in the party selection list box.

The secondary function of the party service is to manage the party data in the form of a local H2 data store using a party event model based file drops for delete and update event files.

At start up the service reads a start-of-day party json file and loads that into the H2 cache. The service then periodically ( 1 sec intervals) sweeps designated event folders for party update and deletion event files. Once events are found they are consumed and applied to the running service h2 cache

For Party events folder location please see the sample service configuration file in Appendix A4.

### The Party Service provides the following rest API for consumption by the WebTool.

### Get Party API

**Protocol** : HTTP and HTTPS

**Method**: GET

**URL** : http://[hostname]:[port]/party/byid?id=1234567890

Request Headers: Content-type : application/json

Accept: application/json

BODY : should contain the JSON structure see Appendix A8 for samples.

**Expected HTTP status response code**

**200** -> OK : Search operation was successfully

**404** -> NOT FOUND : Search operation was not successfully , Party Not Found

**500** -> Server Internal error: indicating that store function failed , check the logs for errors

### Party Query API

**Protocol** : HTTP and HTTPS

**Method**: GET

**URL** : http://[hostname]:[port]/party/query/[--- & delimited list of query parms as name=value *pairs]*

Query Parms:

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type / Format /Value | Sample and Comments |
| term | String | Partial Name search |
| partyIdSerach | String – “true” or “false” | True if contraparty only, false for counterparty only |
| frequentlyUsed | String – “true” or “false” | Return parties designated as  Frequently used |

## 2.3 Operational Data Service Reference implementation

The Reference Data Service is a Java 8 Embedded Jetty HTTP/Jersey based rest API.

The service provides the following reference data rest API.

### FX Rate Quotes API

**Protocol** : HTTP and HTTPS

**Method**: GET

**URL** : http://[hostname]:[port]/refdata/rates

Request Headers: Content-type : application/json

Accept: application/json

BODY : should contain the JSON structure see Appendix A9 for samples.

**Expected HTTP status response code**

**200** -> OK : Search operation was successfully

**500** -> Server Internal error: indicating that store function failed , check the logs for errors

### Participants API

**Protocol** : HTTP and HTTPS

**Method**: GET

**URL** : http://[hostname]:[port]/refdata/participants

Request Headers: Content-type : application/json

Accept: application/json

BODY : should contain the JSON structure see Appendix A9 for samples.

**Expected HTTP status response code**

**200** -> OK : Search operation was successfully

**500** -> Server Internal error: indicating that store function failed , check the logs for errors

### Test User Credentials API

**Protocol** : HTTP and HTTPS

**Method**: GET

**URL** : http://[hostname]:[port]/refdata/users

Request Headers: Content-type : application/json

Accept: application/json

BODY : should contain the JSON structure see Appendix A9 for samples.

**Expected HTTP status response code**

**200** -> OK : Search operation was successfully

**500** -> Server Internal error: indicating that store function failed , check the logs for errors

## 2.4 Dafaulting Data Service Reference implementation

# Appendix

## A.1 General Architecture

../../../Downloads/Droit%20New%20Architecture.pdf

## A.1 Audit Service interfaces and supporting classes

### A.2.1 Audit Service Store Override SPI implementation

Developers of an audit store override SPI must implement the following interface(s)

**public interface** DecisionStoreService {  
  
 */\*\*  
 \* Save a new decision to the underlying data store.  
 \* Caller is responsible for setting the decisionId to a unique value.  
 \** ***@param decision*** *\*/* DecisionRecord saveNewDecision(DecisionRecord decision);  
  
 */\*\*  
 \* Retrieve a single decision bit it's ID.  
 \* All values are accessable except the scenario Analysis File.  
 \** ***@param decisionId*** *\** ***@return*** *\*/* DecisionRecord getById(String decisionId);

}

for Both the saveNewDecision() and getById() methods the GUI will send/expect a DecisionRecord object. The definition of the DecisionRecord is as follows:

**public class** DecisionRecord {  
  
 **private** String **decisionId**;  
 *//@JsonFormat(shape=JsonFormat.Shape.STRING, pattern="yyyy-MM-dd'T'HH:mm:ss.SSS'Z'", timezone="GMT")* **private** Date **decisionDate**; *// in UTC* **private** String **userId**; *// user who created decision from GUI or external source* **private** String **applicationName**; *// application that created decision.* **private** Date **submissionDate**;  
 **private** String **traderId**;  
 **private** String **tradeSalesPersionId**;  
 **private** String **tradeVenue**;  
 **private** String **tradeExternalId**;  
 *//@JsonFormat(shape=JsonFormat.Shape.STRING, pattern="yyyy-MM-dd'T'HH:mm:ss.SSS'Z'", timezone="GMT")* **private** Date **tradeEffectiveDate**;  
 *//@JsonFormat(shape=JsonFormat.Shape.STRING, pattern="yyyy-MM-dd'T'HH:mm:ss.SSS'Z'", timezone="GMT")* **private** Date **tradeTerminationDate**;  
 **private** String **droitDecision**; *// JSON decision* **private** String **tradeAssetClass**;  
 **private** String **tradeBaseProduct**;  
 **private** String **tradeSubProduct**;  
 **private** String **tradeCounterpartyId**;  
 **private** String **tradeContrapartyId**;  
 **private** String **tradeTerm**;  
 **private** BigDecimal **tradeNotional**;  
 **private** Boolean **groupTradeDecision**;  
 **private** Boolean **midRequired**; *// supplement* **private** String **midValue**; *// supplement* **private** Boolean **metRequired**;  
 **private** String **scenarioAnalysisFileName**; *// supplement* **private** String **scenarioAnalysisFileType**; *// supplement* **private** String **scenarioAnalysisFile**; *// supplement* **private** Boolean **override**; *// supplement along with other override fields* **private** String **overrideUser**; *// GUI login that applied the override.  
  
 //@JsonFormat(shape=JsonFormat.Shape.STRING, pattern="yyyy-MM-dd'T'HH:mm:ss.SSS'Z'", timezone="GMT")* **private** Date **overrideDate**; *// time the override was done in UTC* **private** String **overrideComments**;  
 **private** String **overrideApproverId**;  
 **private** String **tradeGoldenSourceId**;  
 **private** String **tradeGroupId**;  
  
 **public** DecisionRecord() {  
  
 }  
  
 **public** String getDecisionId() {  
 **return decisionId**;  
 }  
  
 **public void** setDecisionId(String decisionId) {  
 **this**.**decisionId** = decisionId;  
 }  
  
 **public** DateTime getDecisionDate() {  
 **return new** DateTime(**decisionDate**);  
 }  
  
 **public void** setDecisionDate(DateTime decisionDate) {  
 **this**.**decisionDate** = decisionDate.toDate();  
 }  
  
 **public** DateTime getSubmissionDate() {  
  
 **return new** DateTime(**submissionDate**);  
 }  
  
 **public void** setSubmissionDate(DateTime submissionDate) {  
 **this**.**submissionDate** = submissionDate.toDate();  
 }  
  
 **public** String getTraderId() {  
 **return traderId**;  
 }  
  
 **public void** setTraderId(String traderId) {  
 **this**.**traderId** = traderId;  
 }  
  
 **public** String getTradeSalesPersionId() {  
 **return tradeSalesPersionId**;  
 }  
  
 **public void** setTradeSalesPersionId(String tradeSalesPersionId) {  
 **this**.**tradeSalesPersionId** = tradeSalesPersionId;  
 }  
  
 **public** String getTradeVenue() {  
 **return tradeVenue**;  
 }  
  
 **public void** setTradeVenue(String tradeVenue) {  
 **this**.**tradeVenue** = tradeVenue;  
 }  
  
 **public** String getTradeExternalId() {  
 **return tradeExternalId**;  
 }  
  
 **public void** setTradeExternalId(String tradeExternalId) {  
 **this**.**tradeExternalId** = tradeExternalId;  
 }  
  
 **public** LocalDate getTradeEffectiveDate() {  
  
 **return new** LocalDate(**tradeEffectiveDate**);  
 }  
  
 **public void** setTradeEffectiveDate(LocalDate tradeEffectiveDate) {  
  
 **this**.**tradeEffectiveDate** = tradeEffectiveDate.toDate();  
 }  
  
 **public** LocalDate getTradeTerminationDate() {  
  
 **return new** LocalDate(**tradeTerminationDate**);  
 }  
  
 **public void** setTradeTerminationDate(LocalDate tradeTerminationDate) {  
  
 **this**.**tradeTerminationDate** = tradeTerminationDate.toDate();  
 }  
  
 **public** String getDroitDecision() {  
 **return droitDecision**;  
 }  
  
 **public void** setDroitDecision(String droitDecision) {  
 **this**.**droitDecision** = droitDecision;  
 }  
  
 **public** String getTradeAssetClass() {  
 **return tradeAssetClass**;  
 }  
  
 **public void** setTradeAssetClass(String tradeAssetClass) {  
 **this**.**tradeAssetClass** = tradeAssetClass;  
 }  
  
 **public** String getTradeBaseProduct() {  
 **return tradeBaseProduct**;  
 }  
  
 **public void** setTradeBaseProduct(String tradeBaseProduct) {  
 **this**.**tradeBaseProduct** = tradeBaseProduct;  
 }  
  
 **public** String getTradeSubProduct() {  
 **return tradeSubProduct**;  
 }  
  
 **public void** setTradeSubProduct(String tradeSubProduct) {  
 **this**.**tradeSubProduct** = tradeSubProduct;  
 }  
  
 **public** String getTradeCounterpartyId() {  
 **return tradeCounterpartyId**;  
 }  
  
 **public void** setTradeCounterpartyId(String tradeCounterpartyId) {  
 **this**.**tradeCounterpartyId** = tradeCounterpartyId;  
 }  
  
 **public** String getTradeContrapartyId() {  
 **return tradeContrapartyId**;  
 }  
  
 **public void** setTradeContrapartyId(String tradeContrapartyId) {  
 **this**.**tradeContrapartyId** = tradeContrapartyId;  
 }  
  
 **public** String getTradeTerm() {  
 **return tradeTerm**;  
 }  
  
 **public void** setTradeTerm(String tradeTerm) {  
 **this**.**tradeTerm** = tradeTerm;  
 }  
  
 **public** BigDecimal getTradeNotional() {  
  
 **if** ( **tradeNotional** != **null**)  
 **return tradeNotional**;  
 **else  
 return new** BigDecimal(0);  
 }  
  
 **public void** setTradeNotional(BigDecimal tradeNotional) {  
  
  
 **this**.**tradeNotional** = tradeNotional;  
 }  
  
 **public** String getUserId() {  
 **return userId**;  
 }  
  
 **public void** setUserId(String userId) {  
 **this**.**userId** = userId;  
 }  
  
 **public** String getApplicationName() {  
 **return applicationName**;  
 }  
  
 **public void** setApplicationName(String applicationName) {  
 **this**.**applicationName** = applicationName;  
 }  
  
 **public** Boolean isGroupTradeDecision() {  
 **return groupTradeDecision** != **null** ? **groupTradeDecision** : Boolean.***FALSE***;  
 }  
  
 **public void** setGroupTradeDecision(Boolean groupTradeDecision) {  
 **this**.**groupTradeDecision** = groupTradeDecision;  
 }  
  
 **public** Boolean isMidRequired() {  
 **return midRequired** != **null** ? **midRequired** : Boolean.***FALSE***;  
 }  
  
 **public void** setMidRequired(Boolean midRequired) {  
 **this**.**midRequired** = midRequired;  
 }  
  
 **public** String getMidValue() {  
 **return midValue**;  
 }  
  
 **public void** setMidValue(String midValue) {  
 **this**.**midValue** = midValue;  
 }  
  
 **public** Boolean isMetRequired() {  
 **return metRequired** != **null** ? **metRequired** : Boolean.***FALSE***;  
 }  
  
 **public void** setMetRequired(Boolean metRequired) {  
 **this**.**metRequired** = metRequired;  
 }  
  
 **public** String getScenarioAnalysisFileName() {  
 **return scenarioAnalysisFileName**;  
 }  
  
 **public void** setScenarioAnalysisFileName(String scenarioAnalysisFileName) {  
 **this**.**scenarioAnalysisFileName** = scenarioAnalysisFileName;  
 }  
  
 **public** String getScenarioAnalysisFileType() {  
 **return scenarioAnalysisFileType**;  
 }  
  
 **public void** setScenarioAnalysisFileType(String scenarioAnalysisFileType) {  
 **this**.**scenarioAnalysisFileType** = scenarioAnalysisFileType;  
 }  
  
 **public** String getScenarioAnalysisFile() {  
 **return scenarioAnalysisFile**;  
 }  
  
 **public void** setScenarioAnalysisFile(String scenarioAnalysisFile) {  
 **this**.**scenarioAnalysisFile** = scenarioAnalysisFile;  
 }  
  
 **public** Boolean hasOverride() {  
 **return override** != **null** ? **override** : Boolean.***FALSE***;  
 }  
  
 **public boolean** getOverride() {  
 **return** hasOverride();  
 }  
  
 **public void** setOverride(Boolean override) {  
 **this**.**override** = override;  
 }  
  
 **public** String getOverrideUser() {  
 **return overrideUser**;  
 }  
  
 **public void** setOverrideUser(String overrideUser) {  
 **this**.**overrideUser** = overrideUser;  
 }  
  
 **public** DateTime getOverrideDate() {  
  
 **return new** DateTime(**overrideDate**);  
 }  
  
 **public void** setOverrideDate(DateTime overrideDate) {  
  
 **this**.**overrideDate** = overrideDate.toDate();  
 }  
  
 **public** String getOverrideComments() {  
 **return overrideComments**;  
 }  
  
 **public void** setOverrideComments(String overrideComments) {  
 **this**.**overrideComments** = overrideComments;  
 }  
  
 **public** String getOverrideApproverId() {  
 **return overrideApproverId**;  
 }  
  
 **public void** setOverrideApproverId(String overrideApproverId) {  
 **this**.**overrideApproverId** = overrideApproverId;  
 }  
   
 **public** String getTradeGoldenSourceId() {  
 **return tradeGoldenSourceId**;  
 }  
  
 **public void** setTradeGoldenSourceId(String tradeGoldenSourceId) {  
 **this**.**tradeGoldenSourceId** = tradeGoldenSourceId;  
 }  
  
 **public** String getTradeGroupId() {  
 **return tradeGroupId**;  
 }  
  
 **public void** setTradeGroupId(String tradeGroupId) {  
 **this**.**tradeGroupId** = tradeGroupId;  
 }  
  
}

### A.2.1 Audit Service Query Override SPI implementation

Developers of an audit Query override SPI must implement the following interface(s)

**public interface** DecisionQueryService {  
  
 */\*\*  
 \* Query a list of decision records that match all teh non null values specified in the @DecisionQueryParameters  
 \** ***@param params*** *All non null values will be used to restrict the list of values returned.  
 \* Page number and size are provided to retrieve data in smaller sets.  
 \* Set page size to -1 to get all matching records  
 \** ***@return*** *list of matching decision records.  
 \*/* DecisionQueryResult query(DecisionQueryParameters params);

}

The query() method will take a DecisionQueryParameters object , which contains all of the possible searchable values set by the GUI and must return a DecisionQueryResult object. Also of note the DecicionQueryResult object contains a list of DecisionRecord objects that are the query results.

**public class** DecisionQueryParameters {  
 **private** String **decisionId**;  
 **private** DateTime **decisionDateStart** = **null**;  
 **private** DateTime **decisionDateEnd** = **null**;  
 **private** DateTime **submissionDateStart** = **null**;  
 **private** DateTime **submissionDateEnd** = **null**;  
 **private** Integer **pageNumber** = 0;  
 **private** Integer **pageSize** = 10;  
 **private** String **externalTradeID**;  
 **private** String **groupID**;   
 **private** String **assetClass**;  
 **private** String **product**;  
 **private** String **subProduct**;  
 **private** String **contrapartyID**;  
 **private** String **counterpartyID**;  
 **private** String **trader**;  
 **private** String **salesPerson**;  
 **private** String **userRole**;  
 **private** String **userID**;  
 **private** String **user**;  
 **private** String **overriden**;  
 **private** String **allowedToTrade**;  
 **private boolean canSeeAllTrades**;  
  
 **public** String getDecisionId() {  
 **return decisionId**;  
 }  
  
 **public void** setDecisionId(String decisionId) {  
 **this**.**decisionId** = decisionId;  
 }  
  
 **public** DateTime getDecisionDateStart() {  
 **return decisionDateStart**;  
 }  
  
 **public void** setDecisionDateStart(DateTime decisionDateStart) {  
 **this**.**decisionDateStart** = decisionDateStart;  
 }  
  
 **public** DateTime getDecisionDateEnd() {  
 **return decisionDateEnd**;  
 }  
  
 **public void** setDecisionDateEnd(DateTime decisionDateEnd) {  
 **this**.**decisionDateEnd** = decisionDateEnd;  
 }  
  
 **public** DateTime getSubmissionDateStart() {  
 **return submissionDateStart**;  
 }  
  
 **public void** setSubmissionDateStart(DateTime submissionDateStart) {  
 **this**.**submissionDateStart** = submissionDateStart;  
 }  
  
 **public** DateTime getSubmissionDateEnd() {  
 **return submissionDateEnd**;  
 }  
  
 **public void** setSubmissionDateEnd(DateTime submissionDateEnd) {  
 **this**.**submissionDateEnd** = submissionDateEnd;  
 }  
  
 **public** Integer getPageNumber() {  
 **return pageNumber**;  
 }  
  
 **public void** setPageNumber(Integer pageNumber) {  
 **this**.**pageNumber** = pageNumber;  
 }  
  
 **public** Integer getPageSize() {  
 **return pageSize**;  
 }  
  
 **public void** setPageSize(Integer pageSize) {  
 **this**.**pageSize** = pageSize;  
 }  
  
 **public** String getExternalTradeID() {  
 **return externalTradeID**;  
 }  
  
 **public void** setExternalTradeID(String externalTradeID) {  
 **this**.**externalTradeID** = externalTradeID;  
 }  
  
 **public** String getGroupID() {  
 **return groupID**;  
 }  
  
 **public void** setGroupID(String groupID) {  
 **this**.**groupID** = groupID;  
 }  
  
 **public** String getAssetClass() {  
 **return assetClass**;  
 }  
  
 **public void** setAssetClass(String assetClass) {  
 **this**.**assetClass** = assetClass;  
 }  
  
 **public** String getProduct() {  
 **return product**;  
 }  
  
 **public void** setProduct(String product) {  
 **this**.**product** = product;  
 }  
  
 **public** String getSubProduct() {  
 **return subProduct**;  
 }  
  
 **public void** setSubProduct(String subProduct) {  
 **this**.**subProduct** = subProduct;  
 }  
  
 **public** String getContrapartyID() {  
 **return contrapartyID**;  
 }  
  
 **public void** setContrapartyID(String contrapartyID) {  
 **this**.**contrapartyID** = contrapartyID;  
 }  
  
 **public** String getCounterpartyID() {  
 **return counterpartyID**;  
 }  
  
 **public void** setCounterpartyID(String counterpartyID) {  
 **this**.**counterpartyID** = counterpartyID;  
 }  
  
 **public** String getTrader() {  
 **return trader**;  
 }  
  
 **public void** setTrader(String trader) {  
 **this**.**trader** = trader;  
 }  
  
 **public** String getSalesPerson() {  
 **return salesPerson**;  
 }  
  
 **public void** setSalesPerson(String salesPerson) {  
 **this**.**salesPerson** = salesPerson;  
 }  
  
 **public** String getUserID() {  
 **return userID**;  
 }  
  
 **public void** setUserID(String userID) {  
 **this**.**userID** = userID;  
 }  
  
 **public** String getOverriden() {  
 **return overriden**;  
 }  
  
 **public void** setOverriden(String overriden) {  
 **this**.**overriden** = overriden;  
 }  
  
 **public** String getAllowedToTrade() {  
 **return allowedToTrade**;  
 }  
  
 **public void** setAllowedToTrade(String allowedToTrade) {  
 **this**.**allowedToTrade** = allowedToTrade;  
 }  
  
 **public** String getUser() {  
 **return user**;  
 }  
  
 **public void** setUser(String user) {  
 **this**.**user** = user;  
 }  
  
 **public boolean** canSeeAllTrades() {  
 **return canSeeAllTrades**;  
 }  
  
 **public void** setCanSeeAllTrades(**boolean** canSeeAllTrades) {  
 **this**.**canSeeAllTrades** = canSeeAllTrades;  
 }  
}

**public class** DecisionQueryResult {  
 **private** Collection<DecisionRecord> **decisions** = **new** ArrayList<DecisionRecord>();  
 **private** Integer **totalRecords**;  
 **private** Integer **pageNumber**;  
  
 **public** Collection<DecisionRecord> getDecisions() {  
 **return decisions**;  
 }  
  
 **public void** setDecisions(Collection<DecisionRecord> decisions) {  
 **this**.**decisions** = decisions;  
 }  
  
 **public** Integer getTotalRecords() {  
 **return new** Integer(**decisions**.size());  
 }  
  
 **public void** setTotalRecords(Integer totalRecords) {  
 **this**.**totalRecords** = totalRecords;  
 }  
  
 **public** Integer getPageNumber() {  
 **return pageNumber**;  
 }  
  
 **public void** setPageNumber(Integer pageNumber) {  
 **this**.**pageNumber** = pageNumber;  
 }  
}

## A.3 Sample Reference Data Service yaml configuration file

---  
 **serviceName :** Reference data Service Configuration  
 **version :** 1.0  
  
 *#Service Party Cache and events locations  
 # Can be sourced from the local file system or a MongoDB instance* **partyServiceHostname:** 127.0.0.1  
 **partyServicePort:** 8053  
 **serviceSourceMode:** MongoDb *# FileSystem or MongoDb* **mongoDBHostName:** 192.168.2.20 *# AWS db.droit.local or local office 192.168.2.20* **mongoDBPort:** 27017  
 **mongoDBDatabaseName:** jis *# rds or local .20 jis* **mongoDBCollectionName:** parties *# parties ( rds or local jis )* **partyCacheLocation:** /Users/christopherwhinds/github/referencedataservice/partyCache  
 **partyEventsLocation:** /Users/christopherwhinds/github/referencedataservice/partyEvents  
 **partyStartUpLocation:** /Users/christopherwhinds/github/referencedataservice/partyStartup/  
 **maxPartiesReturnedOnSearch:** 100  
  
 *#Static Data Reference Data Service* **staticDataServicehostname:** 127.0.0.1  
 **staticDataServicePort:** 8054  
 *#Participant json location* **participantLocation:** /Users/christopherwhinds/github/referencedataservice/participants/participants.json  
 *#FX Rates file location* **fxratesLocation:** /Users/christopherwhinds/github/referencedataservice/fxrates/fxRates.json  
 *#User Credentials* **userCredentialsLocation:** //Users/christopherwhinds/github/referencedataservice/users/userCredentials.json  
  
  
 *#Audit Reference Data Servicelocation  
 #Audit Sevices Bindings* **auditServiceHostname:** 127.0.0.1  
 **auditServicePort:** 8052  
 **auditStoreLocation:** /Users/christopherwhinds/github/referencedataservice/auditstore  
 **auditStoreAutoCreateOnStartup:** false

...

## A.4 Reference Data Services Sample log4j2 file

*<?***xml version="1.0" encoding="UTF-8"***?>*<**Configuration status="debug"**>  
 <**Appenders**>  
 <**Console name="CONSOLE" target="SYSTEM\_OUT"**>  
 <**PatternLayout pattern="%d{yyyy-MM-dd HH:mm:ss} %-5p [%t] %c{1}:%L - %m%n"**/>  
 </**Console**>  
 <**File name="SERVICELOGLOG" fileName="../logs/rds-service.log"**>  
 <**PatternLayout pattern="%d{yyyy-MM-dd HH:mm:ss} %-5p [%t] %c{1}:%L - %m%n"**/>  
 </**File**>  
 <**Async name="ASYNC"**>  
 <**AppenderRef ref="SERVICELOGLOG"**/>  
 <**AppenderRef ref="CONSOLE"**/>  
 </**Async**>  
 </**Appenders**>  
 <**Loggers**>  
 <**Logger name="org.apache.log4j.xml" level="all"**/>  
 <**Root level="debug"**>  
 <**AppenderRef ref="ASYNC"**/>  
 </**Root**>  
 </**Loggers**>  
</**Configuration**>

## A.5 Audit Reference Service

Sample decision record in JSON structured format

Store Audit Record API.

This example sends a post to the service to persist the decision record with an ID of **0c8d46bc-c574-4977-86ee-67f5c630ef22XXX**

**HTTP POST**

<http://localhost:8052/audit/store>

**decision payload body**

{

    "decisionId" : "**0c8d46bc-c574-4977-86ee-67f5c630ef22XXX**",

    "decisionDate":"2017-03-17T00:00:00.000+0000",

    "userId":"",

    "applicationName":"",

    "submissionDate":"2017-03-17T00:00:00.000+0000",

    "traderId":"",

    "tradeSalesPersionId":"",

    "tradeVenue":"",

    "tradeExternalId":"",

    "tradeEffectiveDate":"2017-03-17T00:00:00.000+0000",

    "tradeTerminationDate":"2017-03-17T00:00:00.000+0000",

    "tradeAssetClass":"InterestRate",

    "tradeBaseProduct":"IRSwap",

    "tradeSubProduct":"FixedFloat",

    "tradeCounterpartyId":"QHBGK66HXMWOEPOCKW80",

--------------------------- Section Omitted for brevity -----------------------------------------------------------------,

"groupDecision": {

        "workflowSnapshotId": "snapshot.sample.1.0",

        "id": "a21688f7-067e-49f3-80a2-d5724beb6990",

        "marketLogicVersionId": "market.logic.20170316",

        "sef": {

          "tradeAllowed": true,

          "mandated": false

        },

        "businessConduct": {

          "metRequired": false,

          "midRequired": false,

          "tradeAllowed": true,

          "mandated": false

        },

        "clearing": {

          "tradeAllowed": true,

          "mandated": true

        },

        "decisionDate": "2017-03-17T16:52:11.884+0000",

        "tradeAllowed": true,

        "messages": []

      }

    },

    "decisionDate": "2017-03-17",

    "tradeGoldenSourceId": "TMS10000100010101",

    "overrideComments": "NA",

    "decisionId": "0c8d46bc-c574-4977-86ee-67f5c630ef22XXX",

    "tradeEffectiveDate": "2017-03-17",

    "midValue": "NA",

    "metRequired": false,

    "overrideDate": "2017-03-17",

    "overrideApproverId": null,

    "tradeSalesPerson": null,

    "midRequired": false,

    "applicationName": "",

    "tradeSubProduct": "FixedFloat",

    "tradeContrapartyId ": "549300V7NU090M6XCS31",

    "groupTradeDecision": false,

    "userID": null,

    "groupId": null,

    "submissionDate": "2017-03-17",

    "tradeGroupId": "NA",

    "traderId": null,

    "override": false,

    "overrideUser": "NA",

    "tradeTerm": "2D",

    "tradeNotional": 10000000

  }

]

### Query Audit Records API.

This example sends a get request to the service to return a set of decisions that where between entry start date and entry end dates and the trade is “terry trader”.

The response body will return a json array of Decisions that meet the selection criteria.

**HTTP GET**

http://localhost:8052/audit/query?traderName=”terry trader”&decisionDateStart=”2017-01-01”& decisionDateEnd=”2017-01-05”

## A.8 Party Service

### Get Single Party Rest API

**HTTP GET**

http://127.0.0.1:8053/party/byid?id=1234JYTN7D3SW8KCSG26

**RESPONSE BODY**

{

  "CTAInPlace": false,

  "accountDomicileCountryCode": "US",

  "accountID": "1234JYTN7D3SW8KCSG26",

  "accountName": "Master Agreement FX\_Y ISDA\_Y",

  "aifManager": false,

  "alwaysCoveredBondExecution": false,

  "alwaysFileEUEForClient": false,

  "amfDeemedDealer": false,

  "amfDerivativesDealer": false,

  "amfSpecialEntity": false,

  "appropriatenessInfoAndWarnings": false,

  "asicForeignCompany": false,

  "asicReportingDelegate": "N/A",

  "auNexusOptIn": false,

  "usUnclearedStandingEndUserExceptionElection": false,

------------ omitted for brevity ----------------

  "usprUnclearedEndUser": "NotEndUser",

  "voiceDistributionMidVariableMET": true

}

### Query Parties Rest API

**HTTP GET**

http://localhost:8053/party/query?term=Brev&partyIdSearch=true&frequentlyUsed=false

**Response Body**

[

{

"CTAInPlace": false,

"accountDomicileCountryCode": "US",

"accountID": "1GGTTTJYTN7D3SW8KCSG26",

"accountName": "Bank of America",

------------ omitted for brevity ----------------

},

{

"accountDomicileCountryCode": "US",

"accountID": "1234JYTN7D3SW8KCGTDEEE",

"accountName": "Master Card USA",

"aifManager": false,

"alwaysCoveredBondExecution": false,

------------ omitted for brevity ----------------

}

]

## A.6 Operation Data Reference Service

### Participants API

**HTTP GET**

<http://127.0.0.1:8054/refdata/participants>

**PAYLOAD RESPONSE BODY**

[

  {

    "id": "hktrader",

    "name": "Hong Kong Trader",

    "traderlocation": "HK",

    "functions": [

      "TRADER"

    ],

    "email": "devtest@droitfintech.com"

  },

  {

    "id": "autrader",

    "name": "Sidney Trader",

    "traderlocation": "AU",

    "functions": [

      "TRADER"

    ],

    "email": "devtest@droitfintech.com"

  },

  {

    "id": "hksales",

    "name": "Hong Kong Sales",

    "traderlocation": "HK",

    "functions": [

      "SALESPERSON"

    ],

    "email": "devtest@droitfintech.com"

  }

]

## 

### FX Rates Quotes API

**HTTP GET**

<http://127.0.0.1:8054/refdata/rates>

PAYLOAD RESPONSE BODY

[

  {

    "baseCurrency": "AUD",

    "quoteCurrency": "USD",

    "rate": 0.9368,

    "effectiveDate": "2015-12-31"

  },

  {

    "baseCurrency": "BRL",

    "quoteCurrency": "USD",

    "rate": 0.452468,

    "effectiveDate": "2015-12-31"

  },

  {

    "baseCurrency": "CAD",

    "quoteCurrency": "JPY",

    "rate": 92.628,

    "effectiveDate": "2015-12-31"

  },

  {

    "baseCurrency": "EUR",

    "quoteCurrency": "BRL",

    "rate": 2.99,

    "effectiveDate": "2015-12-31"

  },

  {

    "baseCurrency": "EUR",

    "quoteCurrency": "GBP",

    "rate": 0.8,

    "effectiveDate": "2015-12-31"

  }

]

### Boot Strap Test User Credentials API

**HTTP GET**

http://127.0.0.1:8054/refdata/users

PAYLOAD RESPONSE BODY

[

  {

    "username": "trader",

    "role": "ROLE\_TRADER",

    "name": "Terry Trader"

  },

  {

    "username": "admin",

    "role": "ROLE\_ADMIN",

    "name": "Allison Admin"

  },

  {

    "username": "sales",

    "role": "ROLE\_SALESPERSON",

    "name": "Sally Sales"

  },

  {

    "username": "compliance",

    "role": "ROLE\_COMPLIANCE",

    "name": "Carl Compliance"

  },

  {

    "username": "approver",

    "role": "ROLE\_COMPLIANCE",

    "name": "Andy Approver"

  },

  {

    "username": "anup",

    "role": "ROLE\_TRADER",

    "name": "Anup Menon"

}

]