Adept WebTool

Installation and Integration Guide

Release 2.0

Friday, April 14, 2017

Table of Contents

[Overview 3](#_Toc482891572)

[Section 1. WebTool installation 4](#_Toc482891573)

[1.1 System Runtime requirements 4](#_Toc482891574)

[1.1.1 Hardware 4](#_Toc482891575)

[1.1.2 Operating Systems 4](#_Toc482891576)

[1.1.3 Java Runtime Environment (JRE) 4](#_Toc482891577)

[1.1.4 Client Browser 4](#_Toc482891578)

[1.1.5 Application Servers 4](#_Toc482891579)

[1.2 Packaging 4](#_Toc482891580)

[1.3 Installation 6](#_Toc482891581)

[1.4 Basic Configuration – WebTool Properties 6](#_Toc482891582)

[1.5 Logging Configuration 11](#_Toc482891583)

[1.6 Audit Service Plugins 12](#_Toc482891584)

[1.6.1 Durable Enterprise Messaging Decision Store with Rest Query API example. 13](#_Toc482891585)

[Section 2. Reference Data Services – Reference implementation 14](#_Toc482891586)

[2.1 Packaging 14](#_Toc482891587)

[2.2 System Runtime requirements 14](#_Toc482891588)

[2.2.1 Hardware 14](#_Toc482891589)

[2.2.2 Operating Systems 15](#_Toc482891590)

[2.2.3 Java Runtime Environment (JRE) 15](#_Toc482891591)

[2.3 Configure and run the services. 15](#_Toc482891592)

[2.3.1 Configure Service 15](#_Toc482891593)

[2.3.2 Running the service 18](#_Toc482891594)

[2.3.4 Service Logging 19](#_Toc482891595)

[2.4 Source code 19](#_Toc482891596)

[2.5 Audit Service Reference implementation 19](#_Toc482891597)

[Audit Store API 19](#_Toc482891598)

[Audit Query API 19](#_Toc482891599)

[2.6 Party Service Reference Implementation 21](#_Toc482891600)

[The Party Service provides the following rest API for consumption by the WebTool. 21](#_Toc482891601)

[Get Party API 21](#_Toc482891602)

[Party Query API 21](#_Toc482891603)

[2.7 Data Service Reference implementation 22](#_Toc482891604)

[FX Rate Quotes API 22](#_Toc482891605)

[Participants API 22](#_Toc482891606)

[Test User Credentials API 23](#_Toc482891607)

[Appendix 24](#_Toc482891608)

[A.1 General Architecture 24](#_Toc482891609)

[A.2 Audit Service interfaces and supporting classes 24](#_Toc482891610)

[A.2.1 Audit Service Store Override SPI implementation 24](#_Toc482891611)

[A.2.1 Audit Service Query Override SPI implementation 30](#_Toc482891612)

[A.3 Sample WebTool Properties file 33](#_Toc482891613)

[A.4 Sample Reference Data Service yaml configuration file 34](#_Toc482891614)

[A.5 WebTool Logback sample configuration file 35](#_Toc482891615)

[A.6 Reference Data Services Sample log4j2 file 35](#_Toc482891616)

[A.7 Audit Reference Service 35](#_Toc482891617)

[Store Audit Record API 36](#_Toc482891618)

[Query Audit Records API. 38](#_Toc482891619)

[A.8 Party Service 38](#_Toc482891620)

[Get Single Party Rest API 38](#_Toc482891621)

[Query Parties Rest API 39](#_Toc482891622)

[A.9 Reference Data Service 40](#_Toc482891623)

[Participants API 40](#_Toc482891624)

[FX Rates Quotes API 41](#_Toc482891625)

[Boot Strap Test User Credentials API 42](#_Toc482891626)

# Overview

The primary purpose of the document is to describe the installation and integration of the Adept WebTool into a client’s infrastructure. In addition, the document describes how to extend and override basic functionality related the Audit Decision Query and Storage, and implement a custom reliable, fault tolerant, durable and enterprise class solution that suites the client internal IT needs. Section 1.5 covers this topic.

The WebTool ships as a java war JEE container based web application and requires only standard servlet 3.0 container to run. It is agnostic as to which JEE container to use at runtime and is currently tested in an Apache Tomcat7/8 application server. It will work without any problems in other Application Servers like WebSphere ,RedHat JBoss , Oracle WebLogic and Oracle Sun Glassfish.

Webtool Installation is very straightforward requiring minor configuration tweaks to be up and running quickly. In Section 1. installation configuration will be covered in detail.

In section 2 the topic of integration into a client infrastructure is discussed. Reference Data integration is the most important part of the effort of the WebTool into a client’s infrastructure. Out of the box the WebTool ships with a reference implementation of the services that provide the types of json data required by the tool to produce decision results.

The current set of reference data types required that must be provided by clients are;

* Participants, Traders, Sales People sets in JSON format
* Fx Quotes, Fx Quote prices for various currencies sets in JSON format
* Parties, Trading parties, ContraParty and CounterParty sets in JSON format.
* Users, Login users for development and test environments sets in JSON format.

Section 2 will cover these data services and their perspective rest APIs in greater detail.

# Section 1. WebTool installation

## 1.1 System Runtime requirements

### 1.1.1 Hardware

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

* Cores - minimum of 4.
* RAM - Minimum of 8g
* 500 MB disk space.

### 1.1.2 Operating Systems

The WebTool is 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 )
* AIX 6-7X
* Microsoft Windows Server 2012/2016

### 1.1.3 Java Runtime Environment (JRE)

Oracle or Oracle-compatible Java v1.7.55 or higher

### 1.1.4 Client Browser

* Chrome v27 or higher, Chrome is preferred browser by Droit.
* Firefox v20 or higher
* Internet Explorer 9 or lower is NOT supported , Edge not certified.

### 1.1.5 Application Servers

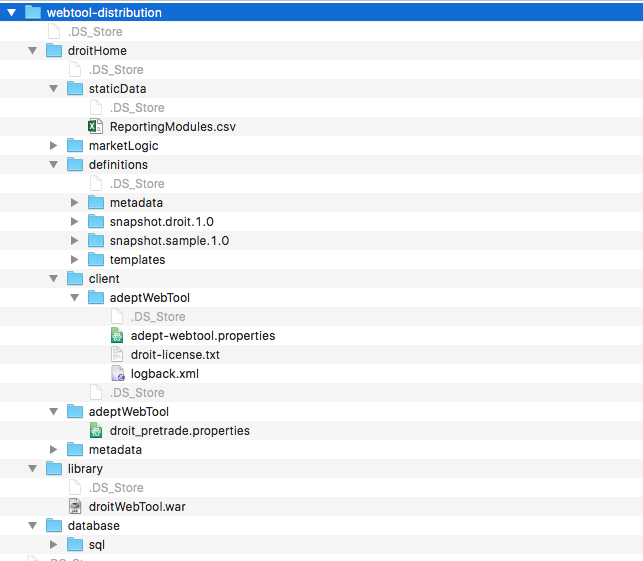
* Tomcat (Default)
* RedHat JBOSS
* IBM WebSphere
* Oracle WebLogic

## 1.2 Packaging

The current software distribution is in the form of a tar.gz file. The file can be retrieved on our Amazon S3 and downloaded landing are at this location:

[TBD](http://xxxxxxxxxxx/XXXXXXXXXXXXXX/XXXXXXXXXXXX/XXXXXXXXXXXXXXXXXXXXXXXXX)

Once Expanded on the file system the distribution folder looks like the following



For the remainder of this document the expanded distribution folder will be referred to as ***$WEBTOOL\_DIST***.

Copy the folder “/droitHome” in the ***$WEBTOOL\_DIST*** folder to another location on the file system. This new “droitHome” folder will become the reference configuration location for the WebTool at runtime.

In subsequent sections of this guide this new folder will be referred to as ***$DROIT\_HOME***.

## 1.3 Installation

This section will describe the installation of the WebTool into a Tomcat Application Server runtime environment.

1. Copy the droitWebTool.war into the tomcat /webapp deployment folder.
2. Modify the Tomcat start script catalina.sh and add the export of the $DROIT\_HOME.

export DROIT\_HOME="/somewhereonthefilesystem/droitHome"

1. Modify catalina.sh script JAVA\_OPTS to include “-Dspring.profiles.active=security.rds” initially

This is the default Reference Data testing only setting. For production, it will switch to Active Directory or production class LDAP as a user credentials source.

4.) To start the service run the tomcat shell script “./startup.sh”.

NOTE: The Configured Tomcat Service cannot be started until all the Reference Data Services, or their equivalent versions provided by the client, are accounted for and running beforehand. The Webtool Application will need to source data from them at startup and throughout service session.

## 1.4 Basic Configuration – WebTool Properties

This table represent the list of configurable WebTool properties options along with both default and optional settings. See Appendix A2 for a sample file definition.

|  |  |  |  |
| --- | --- | --- | --- |
| Configuration Property Name | Required | Allowable values | Description |
| tdss.gui.loginMessage | Optional | String | A text message that will be displayed on the WebTool login Box |
| tdss.utils.email.protocol | Optional | smtp | Protocol used for sending escalation alert messages if configured for notifications |
| tdss.utils.email.host | Optional | Smtp.yourhostname.com | Hostname or IP address of the SMTP server |
| tdss.utils.email.port | Optional | 587 or whatever port used on the server. | SMTP Server port |
| tdss.utils.email.username | Optional |  | Server user name for connection. |
| tdss.utils.email.password. | Optional |  | Server connection password |
| Tdss.utils.email.enabled | Optional | “true” or “false” | Enable or Disable the service |
| adept.workflow.collectionEscalations | Optional |  |  |
| tdss.rules.escalationEmail.subjectPrefix | Optional |  |  |
| tdss.rules.escalationEmail.bodyFooter | Optional |  |  |
| tdss.gui.enableDecisionOverride | Optional | “true” or “false” |  |
| tdss.gui.errorContactEmail | Optional |  |  |
| tdss.gui.enableCustomAttributesPane | Optional | “true” or “false” | Enable or Disable if Custom Party Attributes are in use |
| adept.included.ccps |  |  |  |
| rds.FXRatesQuoteServiceAPI.hostname | Required | IP Address or Hostname | FX Quote Service source Host Name |
| rds.FXRatesQuoteServiceAPI.port | Required | Integer port | FX Quote Service source port |
| rds.FXRatesQuoteServiceAPI.baseUrl | Required | Default /refdata | High level base Url for the FX Quotes service |
| rds.FXRatesQuoteServiceAPI.api | Required | Default /rates | The FXQuotes Service  REST API. |
| rds.ParticipantsServiceAPI.hostname | Required | IP Address or Hostname | Participant Service  Hostname |
| rds.ParticipantsServiceAPI.port | Required | Integer port | Participant Service  Listening port |
| rds.ParticipantsServiceAPI.baseUrl | Required | Default /refdata | High level base Url for the Participants service |
| rds.ParticipantsServiceAPI.api | Required | Default /participants | The Participants Service  REST API. |
| rds.AuditStoreServiceApi.hostname | Optional can be overridden See section 1.5 | IP Address or Hostname | Audit Store Service  Hostname |
| rds.AuditStoreServiceApi.port | Optional can be overridden See section 1.5 | Integer port | Audit Store Service  Listening port |
| rds.AuditStoreServiceApi.baseUrl | Optional can be overridden See section 1.5 | Default /audit | High level base Url for the Audit Store service |
| rds.AuditStoreServiceApi.api | Optional can be overridden See section 1.5 | Default /store | Audit Store service REST API. |
| rds.AuditQueryServiceApi.hostname | Optional can be overridden See section 1.5 | IP Address or Hostname | Audit Query Service  Hostname |
| rds.AuditQueryServiceApi.port | Optional can be overridden See section 1.5 | Integer port | Audit Query Service  Listening port |
| rds.AuditQueryServiceApi.baseUrl | Optional can be overridden See section 1.5 | Default /audit | High level base Url for the Audit Store service |
| rds.AuditQueryServiceApi.api | Optional can be overridden See section 1.5 | Default /query | Audit Store service REST API. |
| rds.PartyGetServiceApi.hostname | Required | IP Address or Hostname | Single Party Get Service  Hostname |
| rds.PartyGetServiceApi.port | Required | Integer port | Single Party Get Service  Listening port |
| rds.PartyGetServiceApi.baseUrl | Required | Default /party | High level base Url for the Single Party Get Service |
| rds.PartyGetServiceApi.api | Required | Default /byid | Audit Store service REST API. |
| rds.PartiesGetServiceApi.hostname | Required | IP Address or Hostname | Query Parties Get Service  Hostname |
| rds.PartiesGetServiceApi.port | Required | Integer port | Query Parties Get Service  Listening port |
| rds.PartiesGetServiceApi.baseUrl | Required | Default /party | High level base Url for the Query Parties Service |
| rds.PartiesGetServiceApi.api | Required | Default /query | Query Parties Service REST API. |
| rds.PartyCountServiceApi.hostname | Required | IP Address or Hostname | Query Party counts Service  Hostname |
| rds.PartyCountServiceApi.port | Required | Integer port | Query Party counts Service  Listening port |
| rds.PartyCountServiceApi.baseUrl | Required | Default /party | High level base Url for the Query Party counts Service |
| rds.PartyCountServiceApi.api | Required | Default /query | Query Party counts Service REST API. |
| tdss.activedirectory.host | Optional | Hostname or IP address/port | Hostname where MS Activie Directory domain controllers is located |
| tdss.activedirectory.domain | Optional | Domain name | Client AD domain name. |
| tdss.activedirectory.searchTerm | Optional | Name of the search name field tag . |  |
| tdss.security.roleMap | Optional | The security Role Map between standard Droit Supported roll names and the client provided role name from Active Directory. | User security role mappings |

## 1.5 Logging Configuration

The WebTool , out of the box, supports logging through the Logback logging API . See Appendix A5 for a sample of the configuration file. For the most part the API does automatic Archiving and supports the standard log levels;

* INFO
* DEBUG
* TRACE;

## 1.6 Audit Service Plugins

Out of the box the Webtool is very flexible as to how it can source reference data(Party , Users, Fx Quotes and Participants) . The current configuration uses HTTP rest calls to any client provided service that returns the required json structured data via specific api signatures and semantics (See Section 2 for details).

The integration of Audit Decision Storage and Query services are a bit more involved. All clients, for compliance purposes, are required to keeping all trade decisions stored and retrievable in a safe and durable fashion. The decision on how to do that is total up to the client as to what best fits the needs of the IT organization.

To accomplish the task of overriding the default Audit Service implementation, the WebTool employs a Service Provider Interface (SPI) strategy. This SPI strategy, often used to override other JVM services like Security Manage and Encryption), will give the client to ability to completely re-implement the mechanisms for storing and retrieving audit records.

To build Override plugins a client must do the following;

1. Implement two classes, one for storing decisions records and one for querying decision records. Both classes must implement specific Droit Java interface contracts;

* *com.droitfintech.tdss.auditservice.store.DecisionStoreService* for the store service
* *com.droitfintech.tdss.auditservice.query.DecsionQueryService* for the query service

1. Compile and build the plugin override classes into a separate jar . Use the distribution jar that contains the above mentioned java interfaces at compile time for resolution of the interfaces and helper class.
2. Provide the new jar on the class path of the container of your choice.
3. Modify the start script of the container of your choice the supply the following override class names at run time ;

* Storage service override use

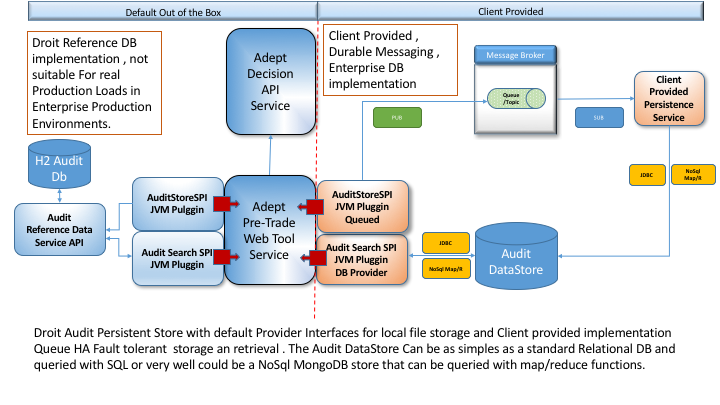
*“-DdecisionStoreSeriveSPI=com.my.DecisionStoreServiceImp”*

* Query Service override use

*“-DdecisionQuerySeriveSPI=com.my.DecisionQueryServiceImp”*

1. The client implementation is responsible for all plugin configuration settings and may use whatever means suitable to accomplish runtime configurations of each plugin.
2. For logging us can use the Logback library or Log4j in the implementation.
3. During plugging initialization must take place at class instanciation time in the constructor.
4. Lastly restart the container so the plugins can be activated.

### 1.6.1 Durable Enterprise Messaging Decision Store with Rest Query API example.



The above image depicts a client implementation of a durable enterprise messaging implementation of the decision storage service which uses a message broker in a publisher/subscriber pattern. The Audit Decision Store service API acts as a proxy to publish the audit decision records to a Client provided persistence storage service, acting as the subscriber, which in turn stores the audit records in whatever facility the client prefers.

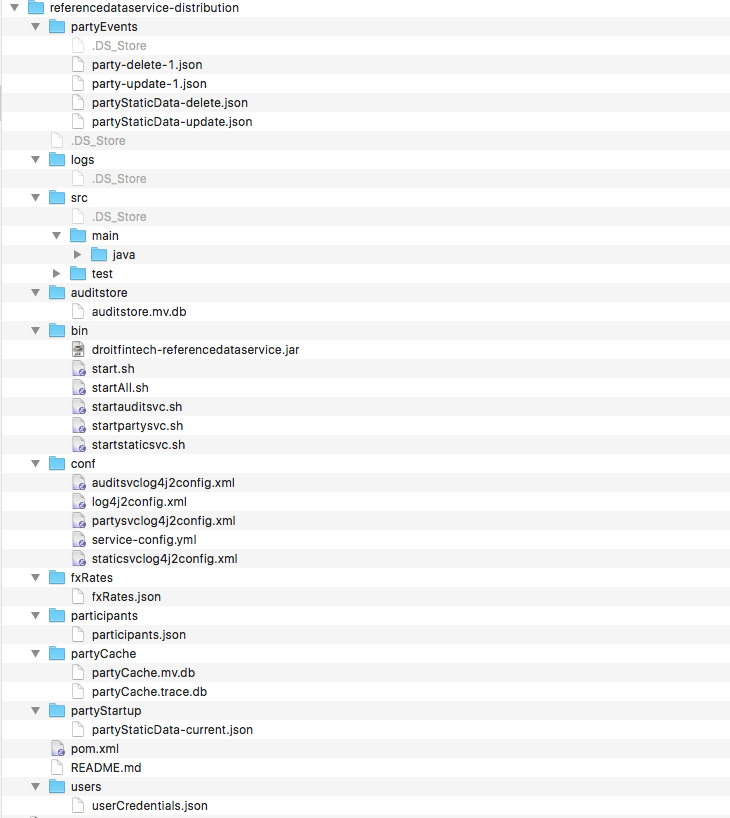
The Audit Query service SPI then queries the storage facility for matching audit records at the request of the GUI service user.

# Section 2. Reference Data Services – Reference implementation

## 2.1 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;



## 2.2 System Runtime requirements

### 2.2.1 Hardware

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

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

### 2.2.2 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 )

### 2.2.3 Java Runtime Environment (JRE)

Oracle or Oracle-compatible Java v1.7.55 or higher and maven 3.3.x and above.

## 2.3 Configure and run the services.

### 2.3.1 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.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Configuration Property Name | Required |  | Allowable Values |  | Comments |  |  |
| **partyServiceHostname** | Required |  | String |  | Hostname or ip address that the service will bind to. |  |  |
| **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. FileSystem is defaualt |  |  |
| **mongoDBHostName** | Optional |  | String |  | Hostname of the MongoDB server |  |  |
| **mongoDBPort** | Optional |  | String |  | MongoDB server Port |  |  |
| **mongoDBDatabaseName** | Optional |  | String |  | MongoDB Database name. |  |  |
| **mongoDBCollectionName** | Optional |  | Strring |  | 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) |  |  |

### 2.3.2 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.

### 2.3.4 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

## 2.4 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.

## 

## 2.5 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

## 2.6 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 |

## 2.7 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

# Appendix

## A.1 General Architecture

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

## A.2 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 WebTool Properties file

*#WebTool Properties  
### Outgoing Mail Properties***tdss.utils.email.protocol**=**smtp  
tdss.utils.email.host**=**smtp.gmail.com  
tdss.utils.email.port**=**587  
tdss.utils.email.username**=**grigorbot@gmail.com  
tdss.utils.email.password**=**Gr1G0rB0t!  
tdss.utils.email.usetls**=**true  
  
tdss.utils.email.enabled**=**true  
  
adept.workflow.collectEscalations**=**true***# Escalation email defaults***tdss.rules.escalationEmail.subjectPrefix**=**Adept Escalation - (${description}) - ${message}  
tdss.rules.escalationEmail.bodyFooter**=**<label>Escalation Data: </label><ul><li><b>Escalation Description: </b>${description}</li><li><b>Message: </b>${message}</li><li><b>Decision ID: </b>${decisionId}</li></ul>***####################################################################### GUI SETTINGS ###***tdss.gui.enableDecisionOverride**=**true  
tdss.gui.errorContactEmail**=**droitSupport@your.bank.here.com  
  
tdss.gui.loginMessage**=**Remember, Bank systems and e-mail must be used in accordance with Bank policies. By using this system, you expressly consent to review of your usage, and content, by the Bank. Access of Bank systems by unauthorized persons is strictly prohibited.  
  
tdss.gui.enableCustomAttributesPane**=**true***####################################################################### CORE ADEPT SETTINGS ###***adept.included.ccps**=**LCH\_CLEARNET\_LTD***#adept.included.sefs=***adept.nrepl.enabled**=**true***####################################################################### JVM Warmup ###  
  
#############################################################################################################  
# Reference Data Service  
#############################################################################################################  
  
# FX Rates API***rds.FXRatesQuoteServiceApi.hostName**=**127.0.0.1  
rds.FXRatesQuoteServiceApi.port**=**8052  
rds.FXRatesQuoteServiceApi.baseUrl**=**/rds  
rds.FXRatesQuoteServiceApi.api**=**/rates***# Participants***rds.ParticipantServiceApi.hostName**=**127.0.0.1  
rds.ParticipantServiceApi.port**=**8052  
rds.ParticipantServiceApi.baseUrl**=**/rds  
rds.ParticipantServiceApi.api**=**/participants***# Audit Store & Query***rds.AuditStoreServiceApi.hostName**=**127.0.0.1  
rds.AuditStoreServiceApi.port**=**8052  
rds.AuditStoreServiceApi.baseUrl**=**/audit  
rds.AuditStoreServiceApi.api**=**/store  
  
rds.AuditQueryServiceApi.hostName**=**127.0.0.1  
rds.AuditQueryServiceApi.port**=**8052  
rds.AuditQueryServiceApi.baseUrl**=**/audit  
rds.AuditQueryServiceApi.api**=**/query***# Party Apis***rds.PartyGetServiceApi.hostName**=**127.0.0.1  
rds.PartyGetServiceApi.port**=**8053  
rds.PartyGetServiceApi.baseUrl**=**/party  
rds.PartyGetServiceApi.api**=**/byid  
  
rds.PartiesServiceApi.hostName**=**127.0.0.1  
rds.PartiesServiceApi.port**=**8053  
rds.PartiesServiceApi.baseUrl**=**/party  
rds.PartiesServiceApi.api**=**/query  
  
rds.PartyCountServiceApi.hostName**=**127.0.0.1  
rds.PartyCountServiceApi.port**=**8053  
rds.PartyCountServiceApi.baseUrl**=**/party  
rds.PartyCountServiceApi.api**=**/cachesize  
  
rds.UserCredentials.hostName**=**127.0.0.1  
rds.UserCredentials.port**=**8052  
rds.UserCredentials.baseUrl**=**/rds  
rds.UserCredentials.api**=**/users***#Active Directory Intgration  
#tdss.activedirectory.host=ldap://oak.fg.rbc.com:389***tdss.activedirectory.domain**=**rbccm.com,sss.com,fg.ss.com,oak.fg.rbc.com  
tdss.activedirectory.host**=**ldap://SEW28616.oak.fg.sss.com:3268  
tdss.activedirectory.searchTerm**=**sAMAccountName***#Spring Security Roles  
#tdss.security.roleMap=DROIT\_Users\_DevQA:ROLE\_STANDARD\_USER  
#tdss.security.roleMap=DROIT\_Users\_DevQA:ROLE\_VIEW\_ALL\_TRADES  
  
  
## Comment this in or out to enable logging of matched market logic rules at DEBUG level***tdss.market-logic-debug**=**true**

## A.4 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.5 WebTool Logback sample configuration file

<**configuration debug="true"**>  
  
 <**appender name="CONSOLE" class="ch.qos.logback.core.ConsoleAppender"**>  
 <**encoder**>  
 <**pattern**>%-5.5level [%10.10thread] %45.45logger{45} : %msg%n</**pattern**>  
 <**immediateFlush**>true</**immediateFlush**>  
 </**encoder**>  
 <**filter class="ch.qos.logback.classic.filter.ThresholdFilter"**>  
 <**level**>WARN</**level**>  
 </**filter**>  
 </**appender**>  
  
 <**appender name="FILE" class="ch.qos.logback.core.rolling.RollingFileAppender"**>  
 <**encoder**>  
 <**pattern**>%date{HH:mm:ss.SSS} %-5.5level [%10.10thread] %45.45logger{45} : %msg%n</**pattern**>  
 <**immediateFlush**>true</**immediateFlush**>  
 </**encoder**>  
 <**file**>/opt/jboss/jboss-eap-6.3/standalone/log/adept.log</**file**>  
 <**rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy"**>  
 <**fileNamePattern**>/opt/jboss/jboss-eap-6.3/standalone/log/archive/adept.log.%d</**fileNamePattern**>  
 <**maxHistory**>30</**maxHistory**>  
 </**rollingPolicy**>  
 </**appender**>  
  
 <**appender name="ASYNC-FILE" class="ch.qos.logback.classic.AsyncAppender"**>  
 <**appender-ref ref="FILE"** />  
 </**appender**>  
  
 <**appender name="ASYNC-CONSOLE" class="ch.qos.logback.classic.AsyncAppender"**>  
 <**appender-ref ref="CONSOLE"** />  
 </**appender**>  
  
 <**logger name="com.droitfintech" level="DEBUG"** />  
  
 *<!--  
 <logger name="com.droitfintech.camelTrace" level="TRACE" />  
 <logger name="com.droitfintech.microservices" level="TRACE" />  
 -->* <**root level="INFO"**>  
 <**appender-ref ref="ASYNC-FILE"** />  
 <**appender-ref ref="ASYNC-CONSOLE"** />  
 </**root**>  
</**configuration**>

## A.6 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.7 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.9 Reference Data 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"

}

]