Nasdaq Calypso

DSMatch Integration Guide

Version 5.5.0

Revision 16.0
February 2024
Approved

Document History

Revision	Published	Summary of Changes
1.0	June 2013	Initial Version
2.0	December 2013	Added changes for November service pack and added legal entity screenshots
3.0	December 2013	Added changes for December service pack
4.0	April 2014	Added changes for April 2014 service pack 1.3.7
5.0	July 2014	Added changes for July 2014 service pack 1.3.8
6.0	August 2014	Added changes for August 2014 service pack 1.3.9
7.0	November 2014	Added changes for November 2014 service pack 1.3.10
8.0	September 2015	Added changes for September 2015 service pack 1.4.0
9.0	March 2019	Added changes for March 2019 service pack 3.3.0
10.0	May 2019	Added changes for May 2019 service pack 3.3.1
11.0	August 2021	Added changes for version 3.4.0
12.0	December 2021	Version 3.5.0 – Technical release only
13.0	January 2022	Version 4.0.0, 4.0.1 - Technical release only – Version 17 compatibility
14.0	August 2022	Version 4.1.0, 3.6.0
15.0	January 2024	Version 5.4.0 - Technical release only – Version 18 compatibility

Revision	Published	Summary of Changes
16.0	February 2024	Version 4.2.0 – Technical release only Version 5.5.0 - – Technical release only

Table of Contents

Release Notes

1.1 Important notes

Please note that the current DSMatch interface supports the FPML-5.3 format only.

1.2 August 2022 – Version 4.1.0

Dependencies

- Calypso 17 September 2022 MR

Component	HD/Case	RPM	Issue	Type	Description
DSMatch 4.1.0		RPM-10844	DSM-194	Sec	Issue – Secure JaxbUnmarshaller. Fix – Addressed security issue with JaxbUnmarshaller.

1.3 August 2022 – Version 3.6.0

Dependencies

- Calypso 16.1 August 2022 MR

Component	HD/Case	RPM	Issue	Type	Description
DSMatch 3.6.0		RPM-10795	DSM-196	Sec	Issue – Secure JaxbUnmarshaller. Fix – Updated Infosec library to version 4.1.19 to address security issue with JaxbUnmarshaller.

1.4 August 2021 – Version 3.4.0

Dependencies

- Calypso 16.1 September 2021 MR

Component	HD/Case	RPM	Issue	Type	Description
DSMatch 3.4.0		RPM-4313	DSM-181	Enh	Issue – Technical issue - Changes for using “internal” Data Uploader module.
DSMatch 3.4.0			DSM-174	Enh	Issue – Technical issue - Remove all instances of e.printStackTrace() and replace with log.

Component	HD/Case	RPM	Issue	Type	Description
DSMatch 3.4.0	172099		DSM-170	Issue	Issue - Trade is created but TradeID and DSMATCHMSG are missing from message report. Already in 3.3.1.
DSMatch 3.4.0			DSM-167	Enh	Issue - Support int and long for the trade ID for the CalypsoTrade and CalypsoTradeKeywords. Already in 3.3.1.

1.5 May 2019 service pack 3.3.1

HD172099 / DSM-169 – Trade is created but TradeID and DSMATCHMSG are missing from message report.

DSM-166 - Support int and long for the trade ID for the CalypsoTrade and CalypsoTradeKeywords.

1.6 March 2019 service pack 3.3.0

HD172098 / DSM-152: Add rule SendNotification to the DSMatch jar.

DSM-164: DSMatch unable to apply action on trade with workflow rule UpdateLinkedToKeyword.

DSM-155: SQL Binding Implementation.

1.7 September 2015 service pack 1.4.0

Please run Execute SQL for DSMatch interface.

HD121925 / DSM-101: Add support for USI/UTI reporting keywords coming in the incoming message as trade keywords and adding the reporting keywords in the acknowledgement sent to DSMatch.

See Support for Reporting Keywords for details.

1.8 November 2014 service pack 1.3.10

HD112234: Trade Allocation support in DSMatch interface in CCP mode: We have fixed an issue reported for a negative test case in the HD for Allocated trades where multiple client/fund legal entities are configured with same DSMatch participant attribute and different clearing brokers as Calypso legal entity parent.

1.9 August 2014 service pack 1.3.9

HD112234: Trade Allocation support in DSMatch interface in CCP mode: We have tested the working of allocation in CCP mode for the DSMatch interface. The fund entity comes as an account reference in the incoming message from DSMatch and needs to be configured as a DSMatchParticipant attribute in one of the calypso legal entity. This

legal entity will be connected to the block or the client entity in Calypso via the PlatformParent attribute. If there are multiple funds configured in calypso with the same BIC code from DSMatch then the corresponding Client entity needs to be also set as a Calypso Legal entity parent on the fund entity.

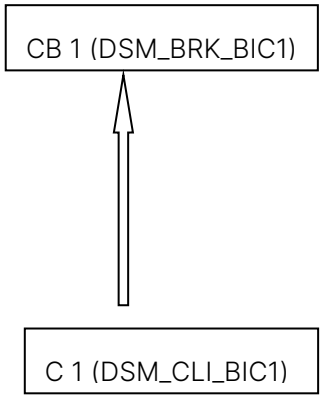
1.10 July 2014 service pack 1.3.8

This is an internal release done for the Calypso version 14 compatibility.

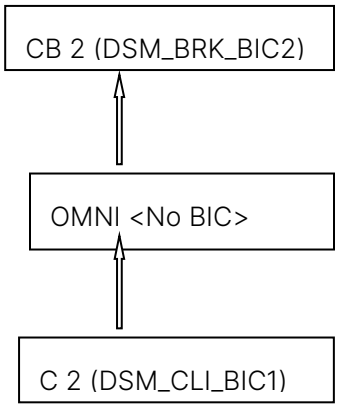
1.11 April 2014 service pack 1.3.7

Please update the latest dsmatchconfig.properties while installing the interface.

HD107125: Client clearing trade booking through DSMatch interface is now supported with the same client setup as segregated with one clearing broker and Omnibus with another clearing broker. We now support multi-level hierarchy for Client-ClearingBroker configuration. In below configuration, if we select DSM_CLI_BIC1 as a client and DSM_BRK_BIC2 as a clearing broker in DSMatch trade then in calypso trade we will have the counterparty as C2 and clearing broker as CB2 respectively.



Segregated Client



Omnibus Client

In the figure above:

- DSM_BRK_BIC1 / DSM_BRK_BIC2 – DSMatch Clearing Broker BIC codes
- DSM_CLI_BIC1 – DSMatch Client BIC code
- C1 / C2 – Calypso legal entities for client
- CB1/CB2 – Calypso legal entities for clearing brokers
- OMNI – Calypso legal entity for Omnibus (No DSMatch BIC associated)

HD107125 / DSM-81: DSMatchOriginalCounterparty trade keyword is now set properly in case the same client is configured as segregated with one clearing broker and Omnibus mode with another clearing broker.

1.12 December 2013 service pack

Changes in December service pack

HD104251 / DSM-76: Incorrect rejection format is getting generated for trades getting rejected at message level. The acknowledgement format for rejections at message level has been changed to ClearingRefused as expected by DSMatch.

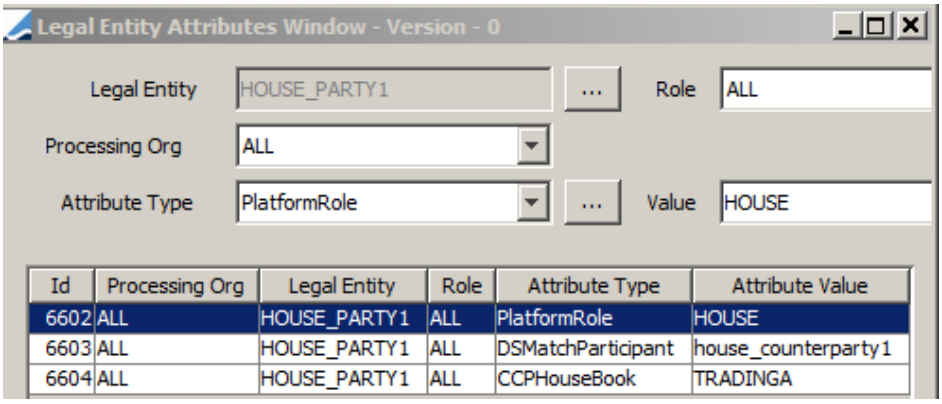
HD102112: Different rejection reason for pair of trade issue. We have resolved this issue in the current service pack.

1.13 November 2013 service pack

Changes in November service pack

HD97848 / DSM-64 / DSM-51: Client clearing trade support. We have added the support for affiliate and client clearing trades submitted through DSMatch. The LE attribute PlatformRole having values HOUSE/CLIENT is introduced at the parent most entity to identify the client versus house clearing. We now the legal entity combinations like:

- Client – Client
- House – House
- Affiliate – Affiliate
- Fund - Fund
- Client - House
- Client – Affiliate
- Client – Fund
- House – Affiliate



Introduction

This document describes the Calypso DSMatch Interface setup. The DSMatch interface is used to clear / declear FX trades.

The interface is comprised of the DSMatch engine which listens to a Message Queue (MQ). It subscribes to Message and Trade events. The events are triggered as soon as a message is posted into the input queue.

The engine is also responsible to send out acknowledgements back to MQ depending on the status / state of the message / trade.

The DSMatch message flows through the configured workflows that route the message to appropriate stages until completion.

The document describes the configuration required to setup the workflows, engine etc. for the DSMatch interface to run successfully.

2.1 Product Coverage

The DSMatch interface supports following FX products:

- FXNonDeliverableForward (FXNDF)

2.2 Software Requirements

2.2.1 Supported JDK Versions

JDK (based on the core Calypso version).

2.2.2 Supported FPML versions

Table 1-1 lists the FPML versions for assets supported by Calypso's

DSMatch interface:

Assets	FPML Version
FX Non Deliverable Forward (FXNDF)	5.3

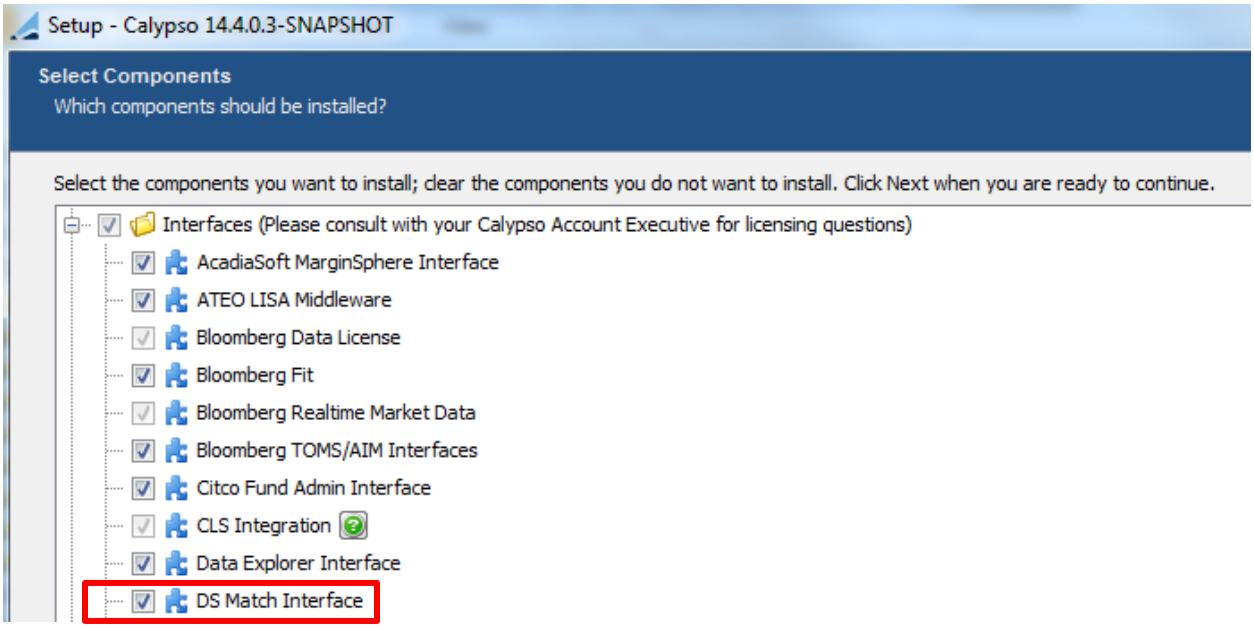
Installation Instructions

3.1 Data Uploader Installation

Refer to the Data Uploader Integration Guide for installation and configuration information. You must install and configure the Calypso Data Uploader prior to configuring the DSMatch. The use of the DSMatch interface requires the Data Uploader.

3.2 Installation

When installing Calypso, select the DS Match Interface in the Calypso Installer.



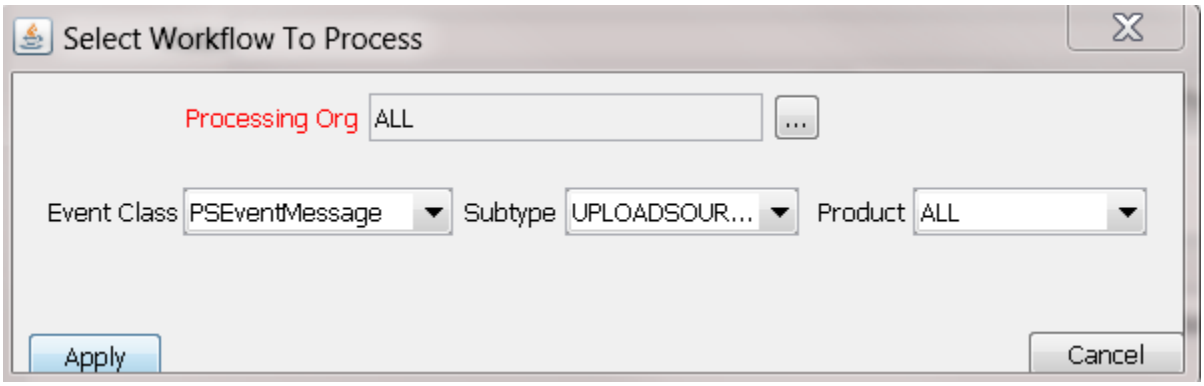
When you run Execute SQL as part of your installation, the DSMatch files will be already loaded. You just need to check “dsmatch” and “gateway” (in addition to you other modules).

Setup Requirements

4.1 Importing Message Workflows

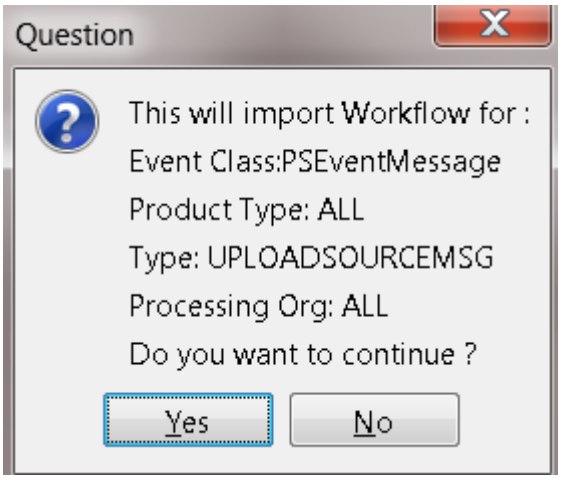
You can import the workflows from the Workflow window: UPLOADSOURCEMSG.wf", "DSMATCHMSG.wf" and "FXNDF.wf". This gives the standard workflow for Gateway BO Messages.

Choose Workflow > Import Current. The following window appears.



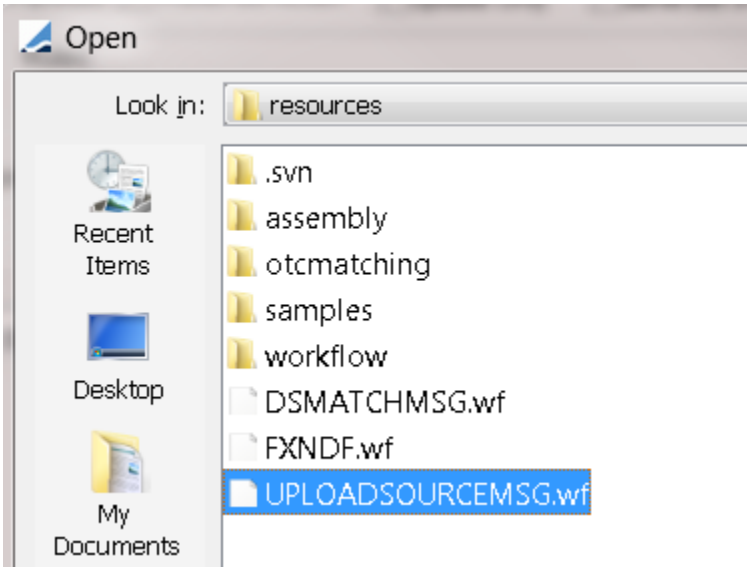
Select the Event class as "PSEventMessage", Subtype as "UPLOADSOURCEMSG" and click **Apply**.

You will be prompted to confirm:

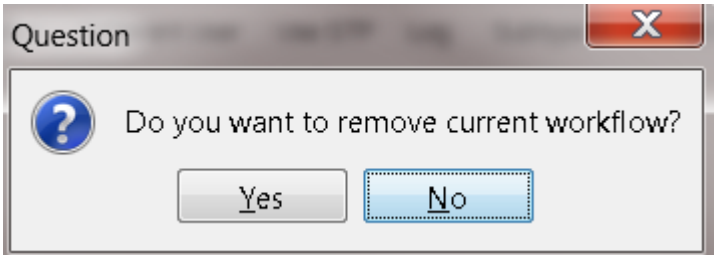


Click **Yes**.

You will then be prompted to select the workflow file:



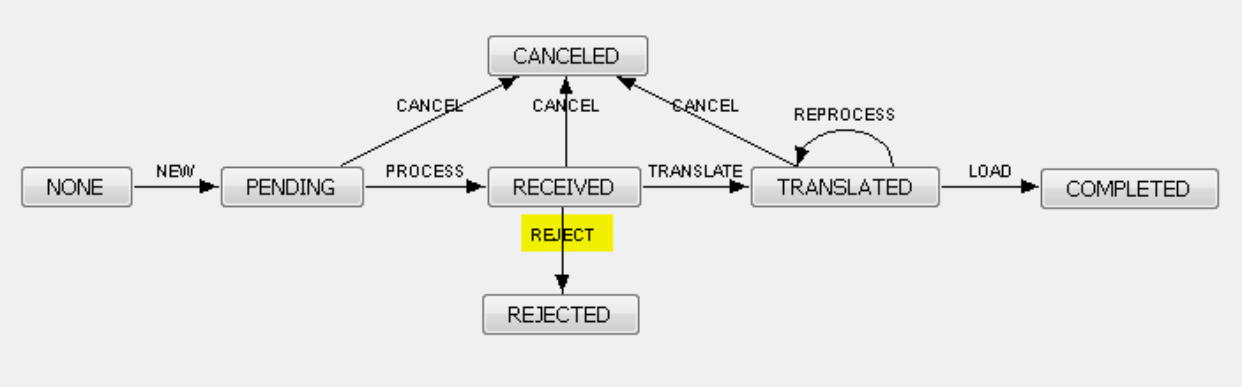
The system prompts to remove the current workflow. Click **No**.



The following window appears indicating that workflow import is successful.

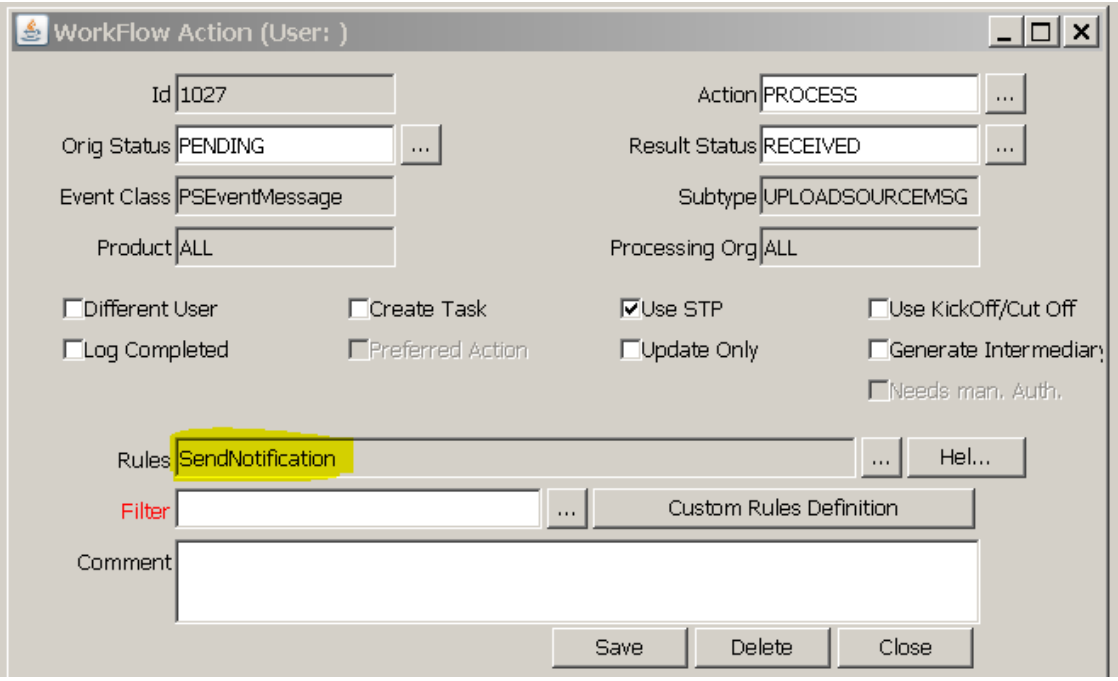
Repeat the above steps for the DSMATCHMSG workflow.

UPLOADSOURCEMSG Workflow

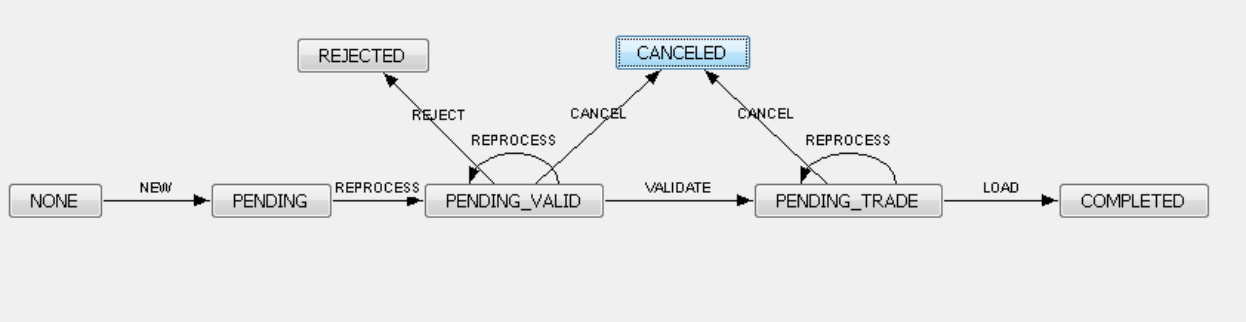


Id	Orig Status	Action	Resulting Status	Different User	Use STP	Log	Subtype	Product Type	Rules	Processing Org	Kick Off/ Cut Off	Create Task
40435	NONE	NEW	PENDING		☑		UPLOADSOURCEMSG	ALL		ALL		
36443	NONE	NEW	RECEIVED				UPLOADSOURCEMSG	ALL		ALL		☑
40436	PENDING	CANCEL	CANCELED				UPLOADSOURCEMSG	ALL		ALL		
40437	PENDING	PROCESS	RECEIVED		☑		UPLOADSOURCEMSG	ALL	SendNotification	ALL		
36444	RECEIVED	CANCEL	CANCELED				UPLOADSOURCEMSG	ALL		ALL		
36445	RECEIVED	TRANSLATE	TRANSLATED		☑		UPLOADSOURCEMSG	ALL	CheckLink,Translate	ALL		☑
40438	TRANSLATED	CANCEL	CANCELED				UPLOADSOURCEMSG	ALL		ALL		
36446	TRANSLATED	LOAD	COMPLETED		☑		UPLOADSOURCEMSG	ALL	CreateUploadMessage	ALL		☑
36447	TRANSLATED	REPROCESS	RECEIVED				UPLOADSOURCEMSG	ALL		ALL		
40439	TRANSLATED	REPROCESS	TRANSLATED				UPLOADSOURCEMSG	ALL	Translate	ALL		

Make sure that the workflow rule SendNotification is added on the transition PENDING – PROCESS – RECEIVED:



DSMATCHMSG Workflow

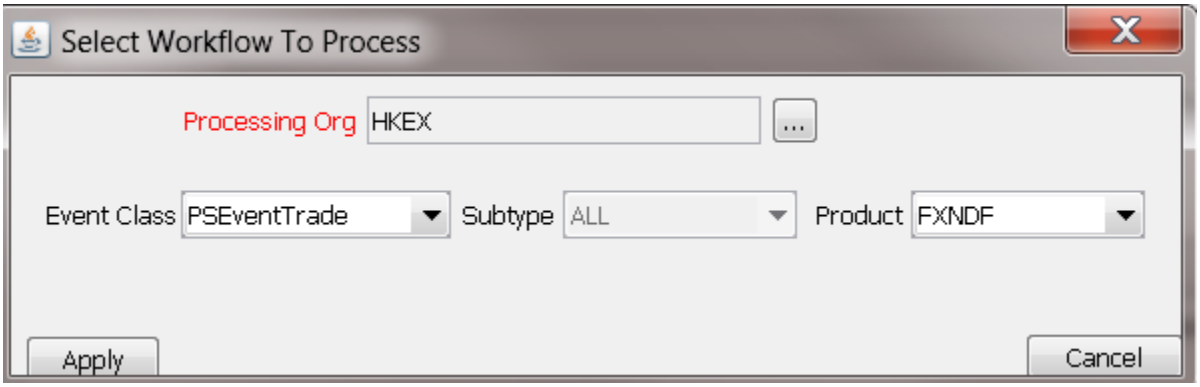


Id	Orig Status	Action	Resulting Status	Different User	Use STP	Log	Subtype	Product Type	Rules	Processing Org	Kick Off/ Cut Off
40440	NONE	NEW	PENDING		☑		DSMATCHMSG	ALL		ALL	
36435	NONE	NEW	PENDING_VALID		☑		DSMATCHMSG	ALL		ALL	
40441	PENDING	REPROCESS	PENDING_VALID		☑		DSMATCHMSG	ALL	UpdateLinkedToAttribute	ALL	
36436	PENDING_TRADE	CANCEL	CANCELED				DSMATCHMSG	ALL		ALL	
36437	PENDING_TRADE	LOAD	COMPLETED		☑		DSMATCHMSG	ALL	CheckLink,Loader	ALL	
36438	PENDING_TRADE	REPROCESS	PENDING_TRADE				DSMATCHMSG	ALL	ReMap	ALL	
36439	PENDING_VALID	CANCEL	CANCELED				DSMATCHMSG	ALL		ALL	
36440	PENDING_VALID	REJECT	REJECTED		☑		DSMATCHMSG	ALL	ApplyLinkedMsgAction,NotValidGatewayMSG	ALL	
36441	PENDING_VALID	REPROCESS	PENDING_VALID				DSMATCHMSG	ALL	ReMap	ALL	
36442	PENDING_VALID	VALIDATE	PENDING_TRADE		☑		DSMATCHMSG	ALL	Validate	ALL	

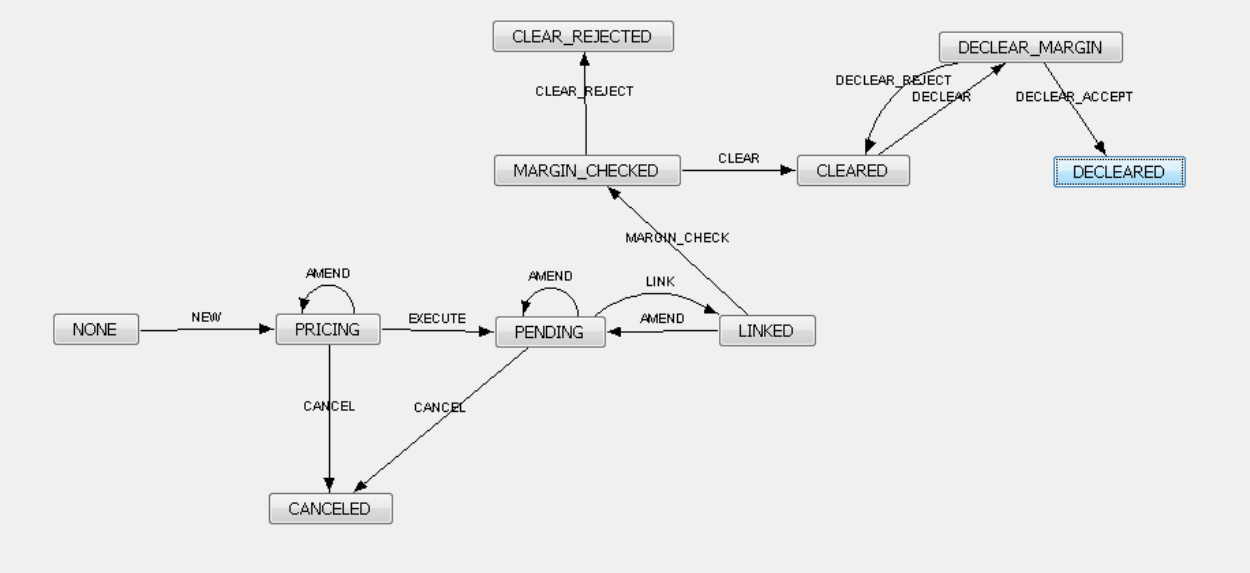
4.2 Importing Trade Workflow

Please make sure that you have the processing org HKEX already setup.

For the trade workflow, the first step changes as shown in the screenshot below:



The FXNDF trade workflow: Make sure all the rules are present and the PO is HKEX (The one that has book mapping) is also configured.



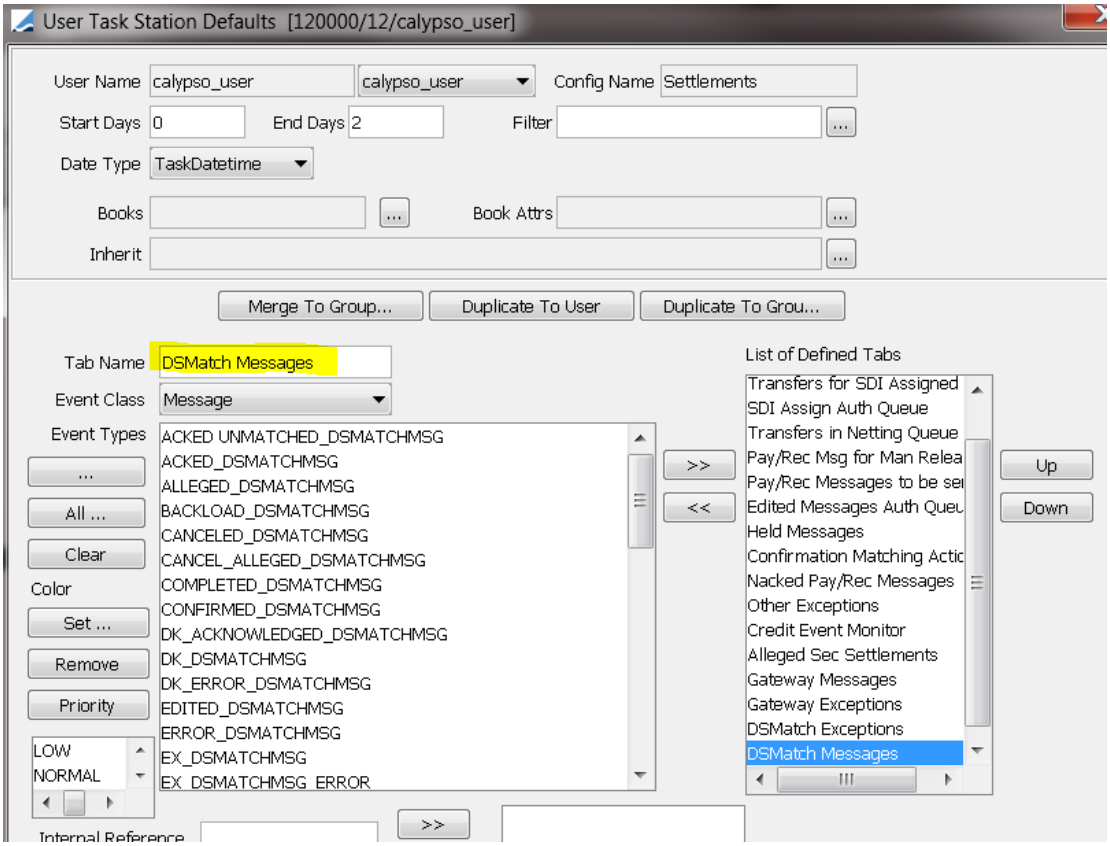
Id	Orig Status	Action	Resulting Status	Different User	Use STP	Log	Subtype	Product Type	Rules	Processing Org	Kick Off/ Cut Off
37435	CLEARED	DECLEAR	DECLEAR_MARGIN				ALL	FXNDF		HKEX	
37436	DECLEAR_MARGIN	DECLEAR_ACCEPT	DECLEARED				ALL	FXNDF	ApplyLinkedTradeAction,DeclearAction	HKEX	
37437	DECLEAR_MARGIN	DECLEAR_REJECT	CLEARED				ALL	FXNDF	ApplyLinkedTradeAction	HKEX	
37438	LINKED	AMEND	PENDING				ALL	FXNDF	ApplyLinkedTradeAction	HKEX	
37439	LINKED	MARGIN_CHECK	MARGIN_CHECKED				ALL	FXNDF	ApplyLinkedTradeAction	HKEX	
37440	MARGIN_CHECKED	CLEAR	CLEARED				ALL	FXNDF	ApplyLinkedTradeAction	HKEX	
37441	MARGIN_CHECKED	CLEAR_REJECT	CLEAR_REJECTED				ALL	FXNDF	ApplyLinkedTradeAction	HKEX	
37442	NONE	NEW	PRICING				ALL	FXNDF		HKEX	
37443	PENDING	AMEND	PENDING				ALL	FXNDF		HKEX	
37444	PENDING	CANCEL	CANCELED				ALL	FXNDF	Cancel	HKEX	
37445	PENDING	LINK	LINKED		✓		ALL	FXNDF	UpdateLinkedToKeyword	HKEX	
37446	PRICING	AMEND	PRICING				ALL	FXNDF		HKEX	
37447	PRICING	CANCEL	CANCELED				ALL	FXNDF		HKEX	
37448	PRICING	EXECUTE	PENDING		✓		ALL	FXNDF	UpdateLinkedToKeyword	HKEX	

4.3 Task Station Setup

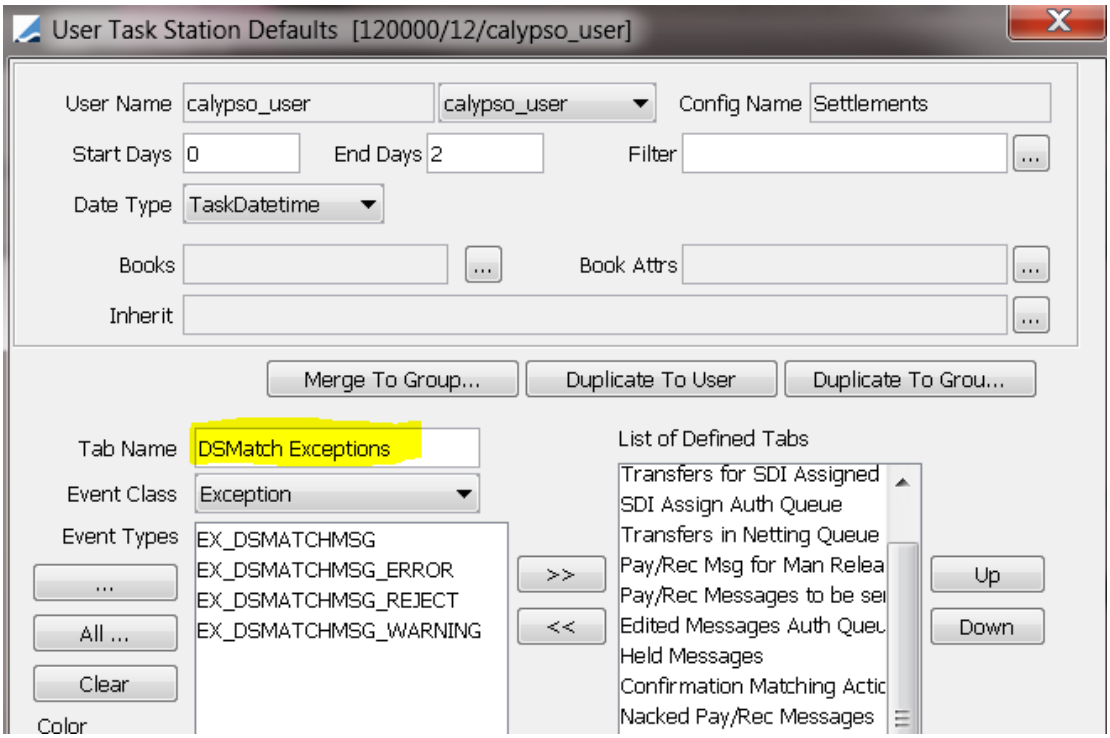
The DSMatch interface creates task station entries for all the messages and exceptions that are encountered. The user can view / reprocess the messages that are failed in validation from the task station.

Add DSMatchMessages and DSMatchExceptions tabs in the Task Station as shown below.

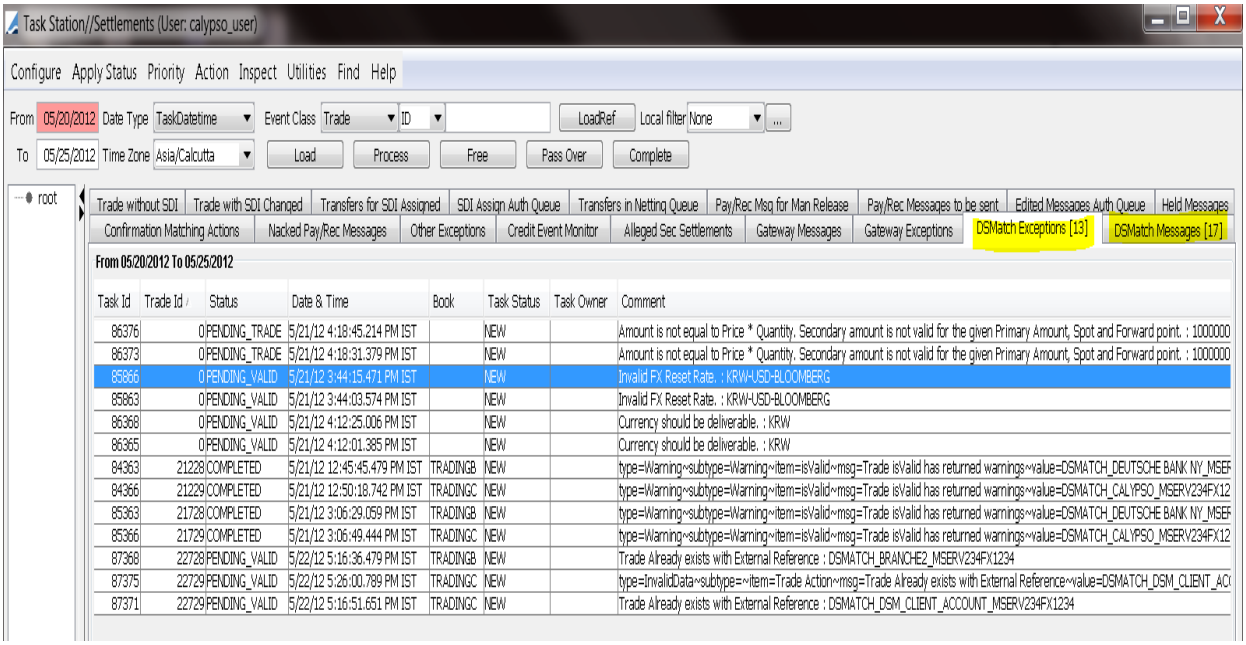
DSMatch Messages



DSMatch Exceptions



The exceptions can be viewed in the task station. The below screen shot displays exceptions:

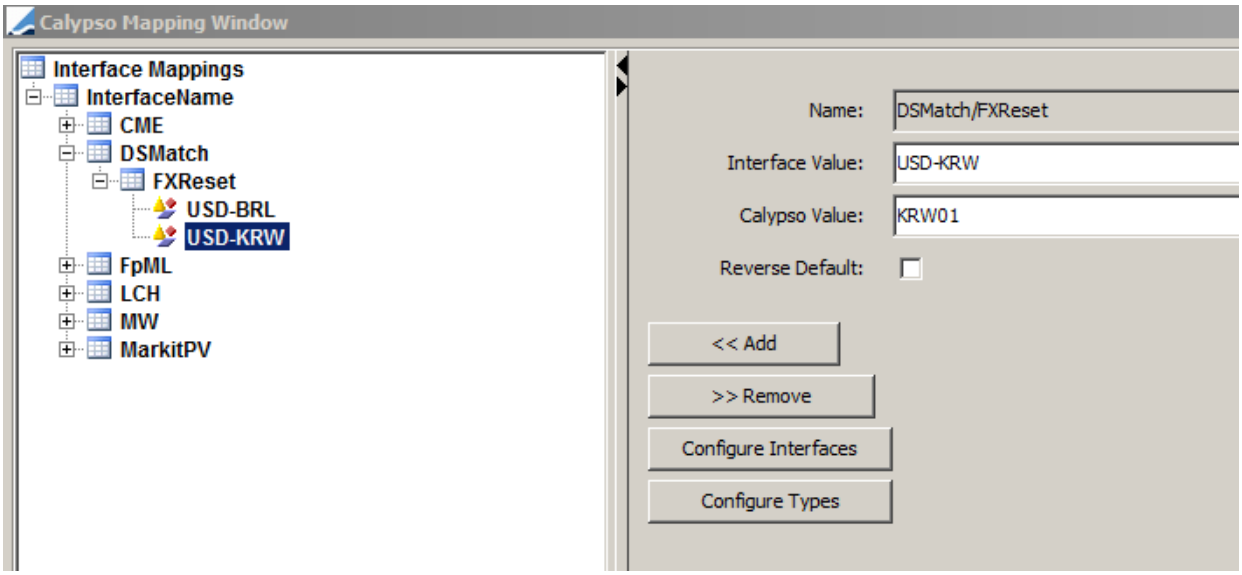


4.4 Calypso Mapping Window Setup

Mapping from Calypso values to the MarkitServ-DSMatch values.

All the columns / types of calypso that have different values at MarkitServ-DSMatch side need to be mapped with corresponding calypso values from the Calypso Mapping Window (menu action `mapping.CalypsoMappingWindow`).

Add the mapping for the currency pair and the fx-reset name. The Mapping window will look like as shown below:



4.5 DSMatch Engine Setup

The DSMatch engine is responsible for both getting messages from sender MQ and also sending back the acknowledgements to the recipient MQ.

The engine handles the Message and Trade events.

4.5.1 Setup the Queue Config File

If the config name passed to the engine is set to “DSMatchFX”, then there should be a config file existing with the name “calypso_dsmatchfx_config.properties”.

This file contains the properties used to identify the input and output queues. The following are the properties in the config file:

```
# DSMatch Import Message Engine (MQ)

#Inbound queue details
input.queue.name=dynamicQueues/InputQ
dynamicQueues/InputQ.queue.setContext=true

#For ActiveMQ
jms.url=tcp://localhost:61616
jms.modetypeclass=org.apache.activemq.jndi.ActiveMQInitialContextFactory
jms.queue.connectionFactory=ConnectionFactory

#Outbound queue details
output.queue.name=dynamicQueues/OutputQ
dynamicQueues/OutputQ.queue.ackType=auto
dynamicQueues/OutputQ.queue.persist=true
dynamicQueues/OutputQ.queue.transacted=false
```

4.5.2 Configure the properties for 2 CCP-ID Mode

We can configure the DSMatch interface to work with 2 CCP ID generation in the acknowledgements or single Id depending upon the requirements.

The property file to edit for this is “dsmatchconfig.properties”.

The below is the snapshot of the property file. We can set the “twoCCPIdMode” flag as true/false as per the requirement.

```
# Map DSMatch products to Calypso Identifers.
# DO NOT MODIFY.
fxNDF=FXNDF

# This flag is used to generate XML files from the FPML Messages. Two XML files are generated for every FPML
# message. One file is the FPML message itself, the other is Calypso Data Upload XML (which is the Calypso
# representation of the FPML. Please refer to documentation for more details.
# The files are generated in USER_HOME\Calypso\dsmatch
DEBUG_XML=true

# This flag indicates the 2 ccp id to be sent in the acks if true, else the engine will send only 1 ccp id
twoCCPIdMode=false
```

Please note that the “DEBUG_XML” property can be set if we need the incoming messages to be archived for debugging / retesting purposes.

4.5.3 Configuring DSMatch Engine

The DS Match engine is configured in the Engine Manager of Web Admin: event subscription and engine parameters.

You may need to add this engine if it is not available for configuration: Create a new engine called DSMatchFXDSMatchImportMessageEngine with class name `com.calypso.tk.engine.`ImportMessageEngine.

It subscribes to PSEventTrade and PSEventMessage events.

The "config" engine parameter is config = DSMatchFX

Filter = DSMatchImportEngineEventFilter

The DSMatch engine can be started from the Engine Manager in Web Admin.

▶ Please refer to Calypso Web Admin documentation for complete details.

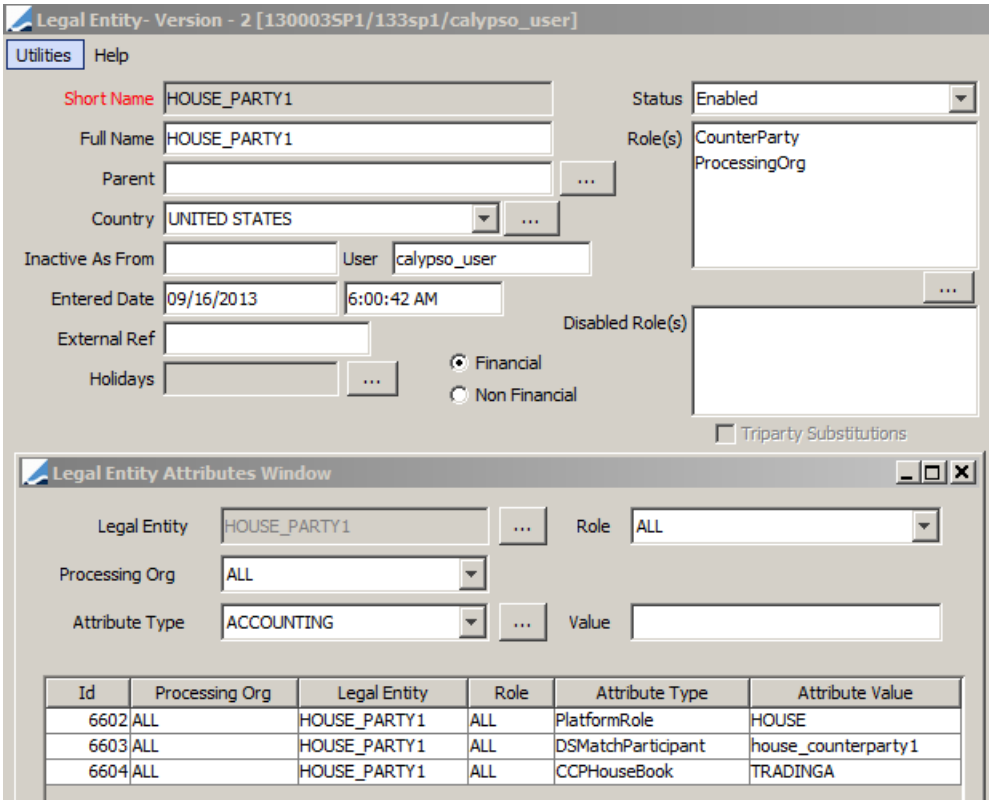
4.6 Legal Entity Mapping for Member

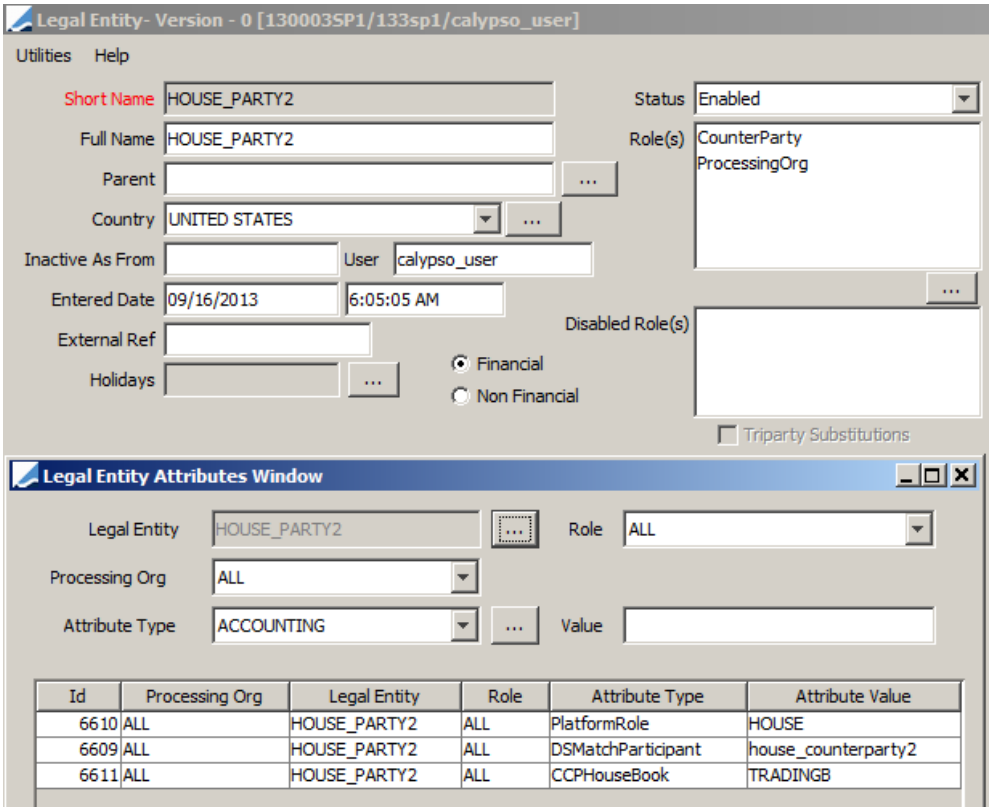
Given the party information, the Legal Entity (Counterparty) will be identified in calypso using the Legal Entity attribute DSMatchParticipant and the value would match the party ID given in FPML.

The fetching logic will first search for the attribute and if not found then it will search for an entity having a matching shortname.

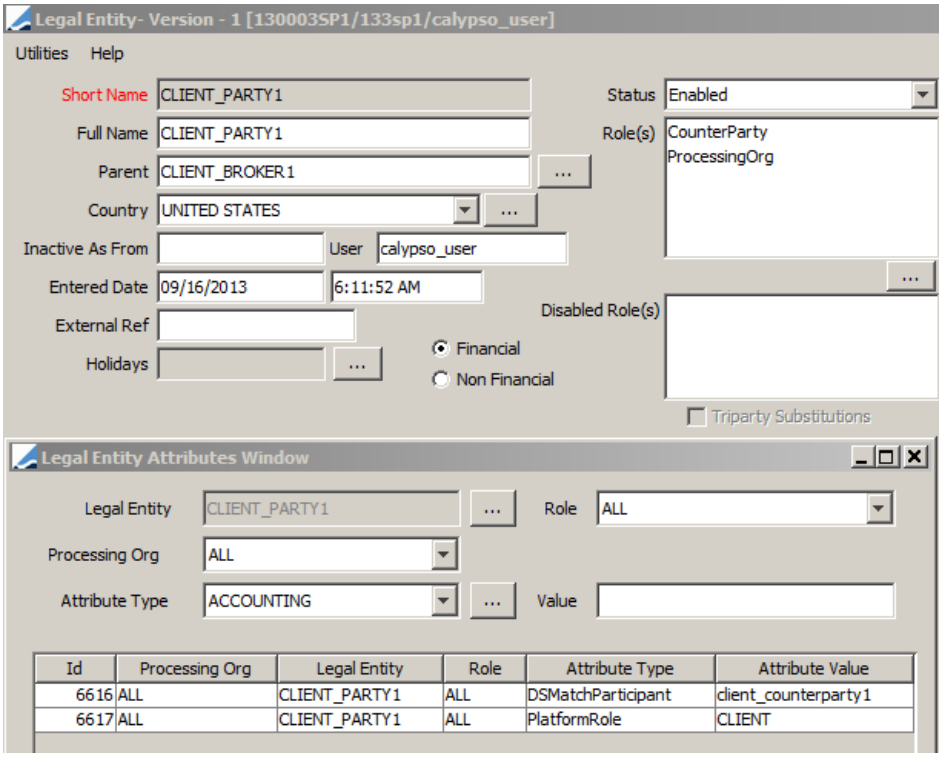
The PlatformRole attribute identifies the LE being in hierarchy of HOUSE / CLIENT.

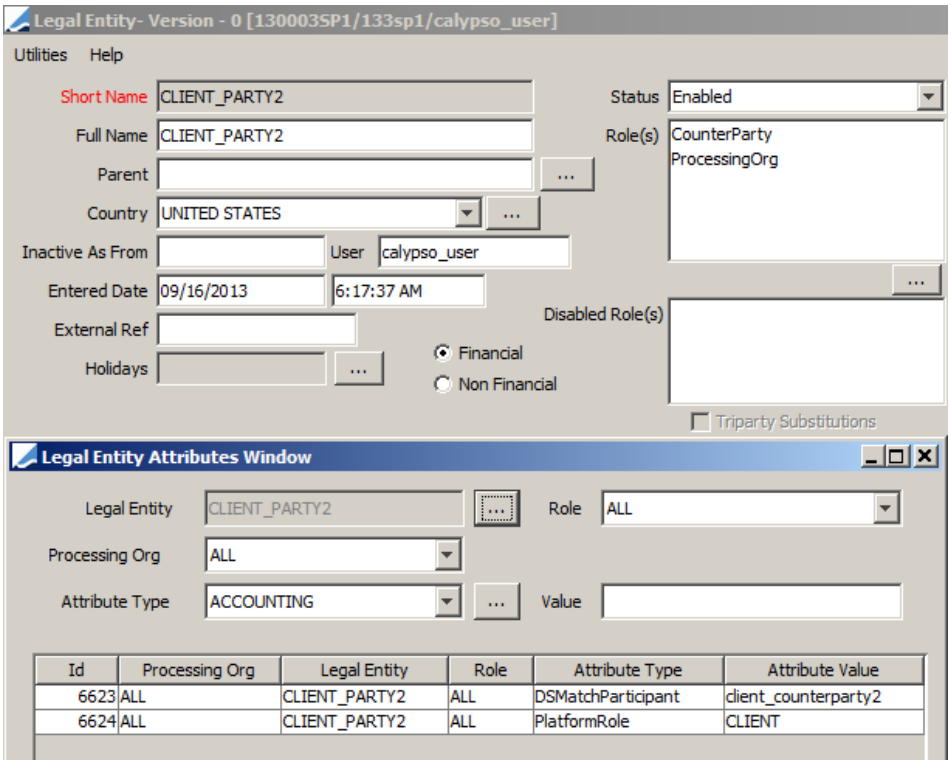
HOUSE Entities:


Legal Entity- Version - 2 [130003SP1/133sp1/calypso_user]
Utilities
Help
Short Name
HOUSE_PARTY1
Status
Enabled
Full Name
HOUSE_PARTY1
Role(s)
CounterParty
ProcessingOrg
Parent
Country
UNITED STATES
Inactive As From
User
calypso_user
Entered Date
09/16/2013
6:00:42 AM
External Ref
Holidays
Financial
Non Financial
Disabled Role(s)
Triparty Substitutions
Legal Entity Attributes Window
Legal Entity
HOUSE_PARTY1
Role
ALL
Processing Org
ALL
Attribute Type
ACCOUNTING
Value
Id
Processing Org
Legal Entity
Role
Attribute Type
Attribute Value
6602
ALL
HOUSE_PARTY1
ALL
PlatformRole
HOUSE
6603
ALL
HOUSE_PARTY1
ALL
DSMatchParticipant
house_counterparty1
6604
ALL
HOUSE_PARTY1
ALL
CCPHouseBook
TRADINGA

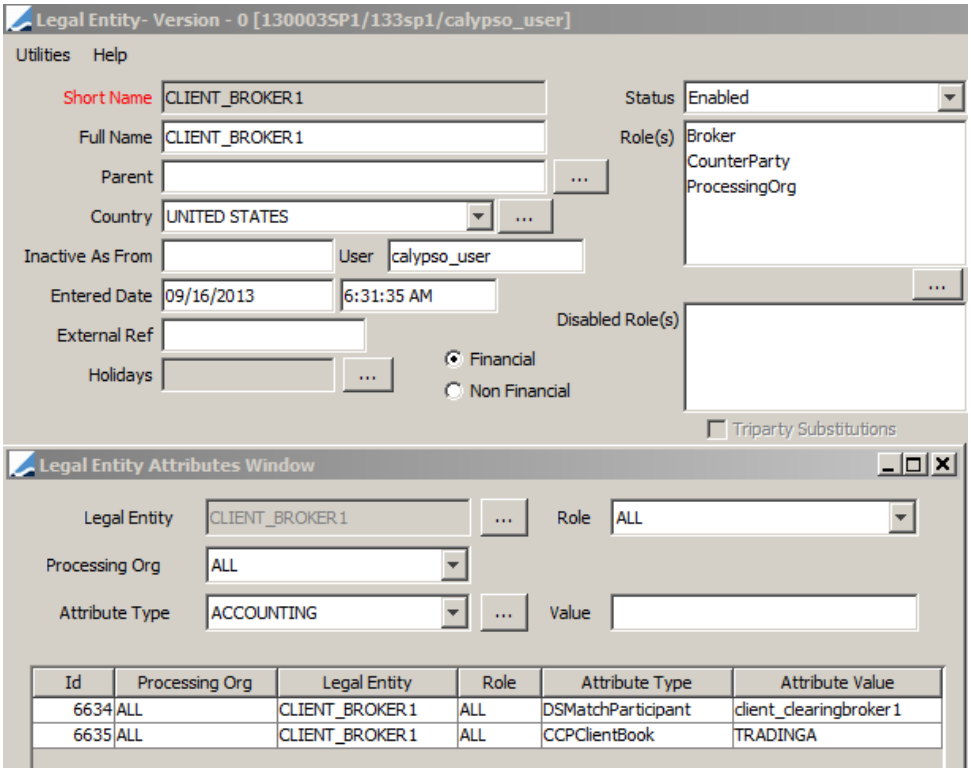

Legal Entity- Version - 0 [130003SP1/133sp1/calypso_user]
Utilities
Help
Short Name
HOUSE_PARTY2
Status
Enabled
Full Name
HOUSE_PARTY2
Role(s)
CounterParty
ProcessingOrg
Parent
Country
UNITED STATES
Inactive As From
User
calypso_user
Entered Date
09/16/2013
6:05:05 AM
External Ref
Holidays
Financial
Non Financial
Disabled Role(s)
Triparty Substitutions
Legal Entity Attributes Window
Legal Entity
HOUSE_PARTY2
Role
ALL
Processing Org
ALL
Attribute Type
ACCOUNTING
Value
Id
Processing Org
Legal Entity
Role
Attribute Type
Attribute Value
6610
ALL
HOUSE_PARTY2
ALL
PlatformRole
HOUSE
6609
ALL
HOUSE_PARTY2
ALL
DSMatchParticipant
house_counterparty2
6611
ALL
HOUSE_PARTY2
ALL
CCPHouseBook
TRADINGB

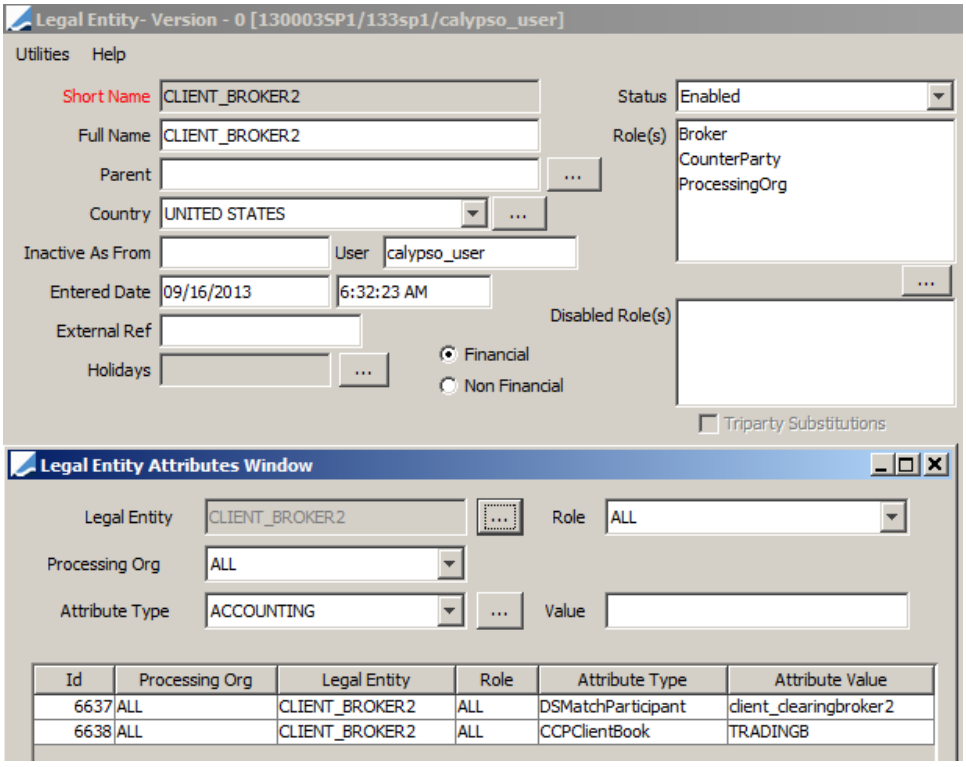
4.7 Legal Entity Mapping for Client





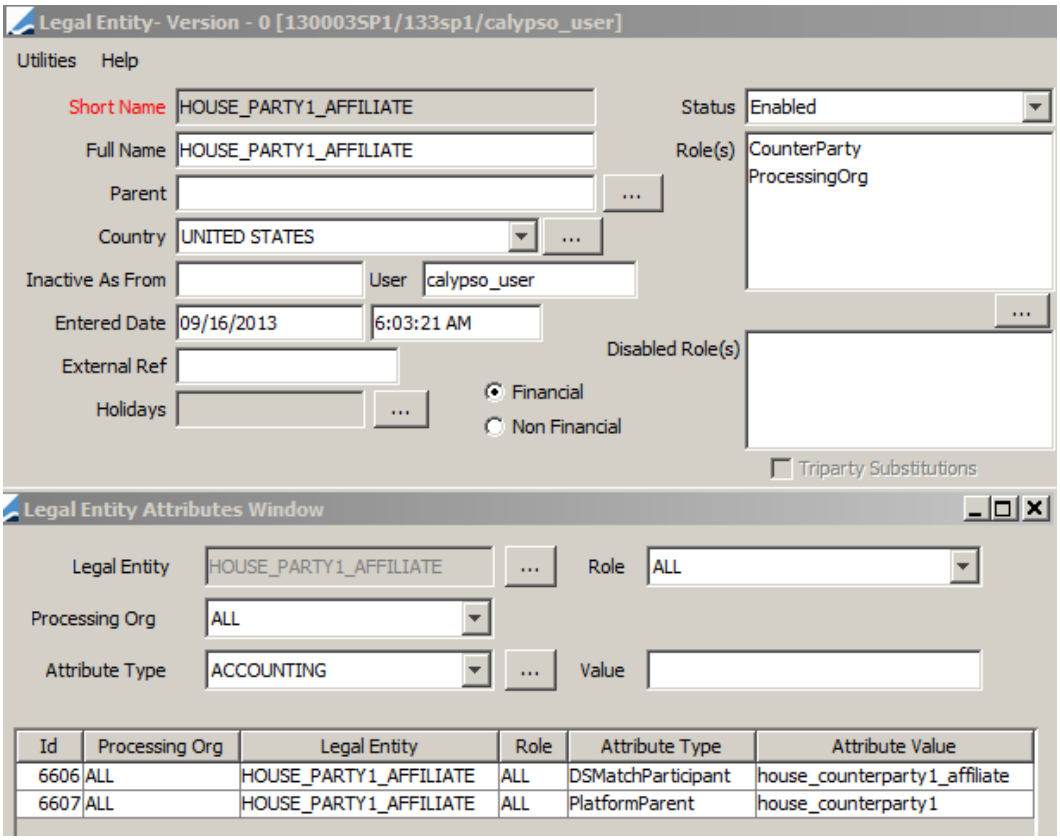
4.8 Legal Entity Mapping for Broker



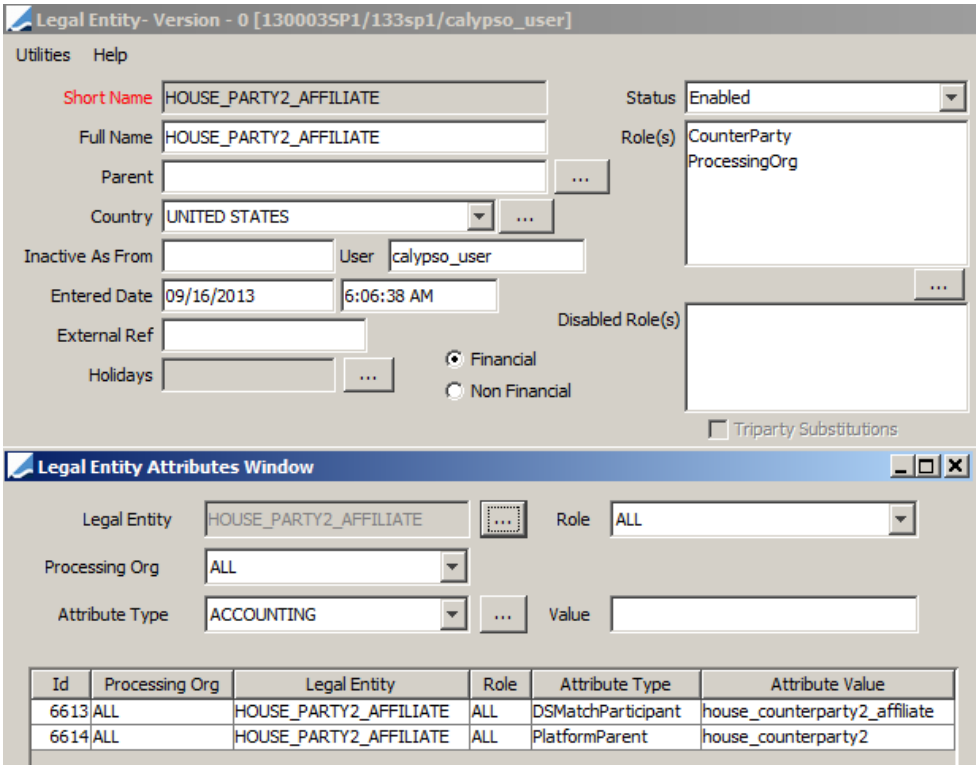


4.9 Legal Entity Mapping for Affiliates

Add the DSMatchParent attribute to identify the legal entity code of the affiliate.

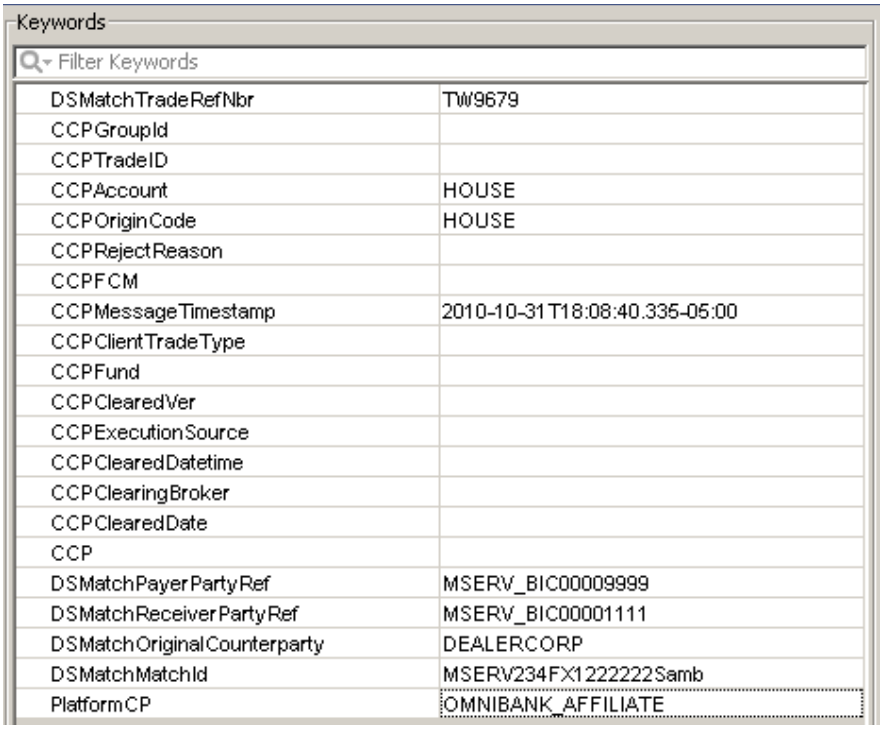


Similarly, you can cofigure for the other party.

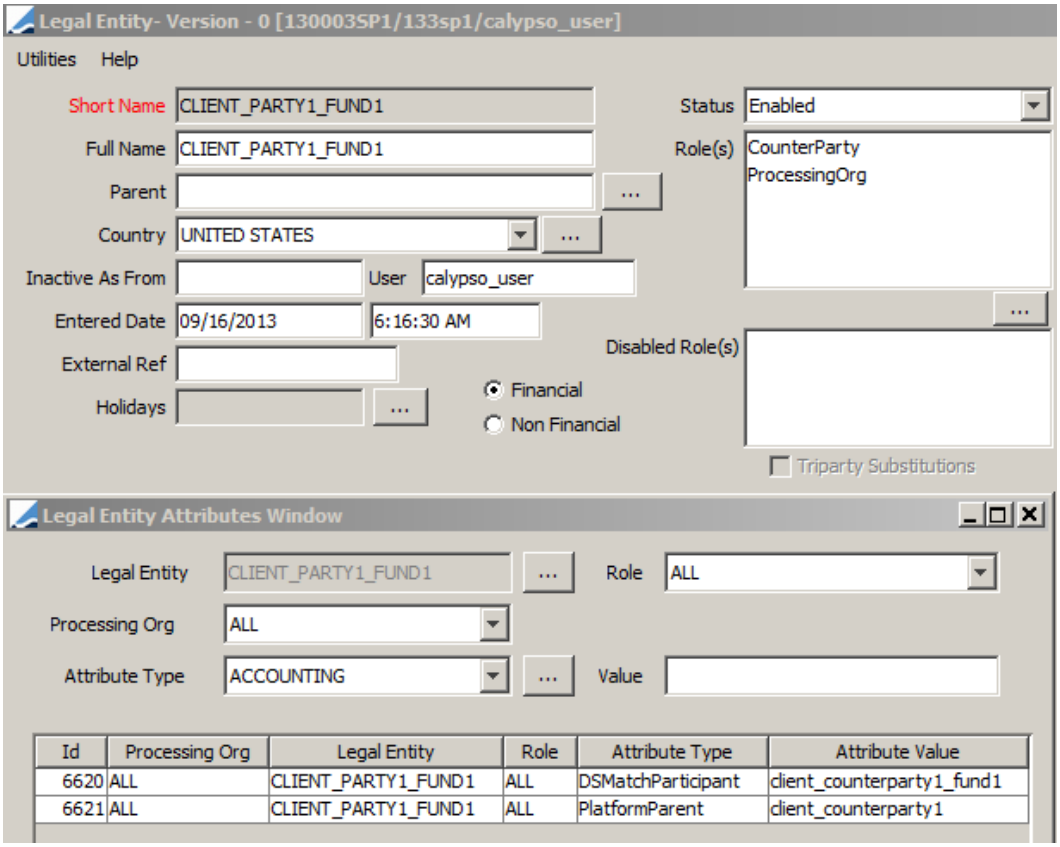


If one of the parties in the trades is an Affiliate entity, then the parent entity is stored as the trade counterparty and the affiliate entity is stored in a trade keyword – "PlatformCP".

Please refer the screenshot below, showing the keyword on the trade:



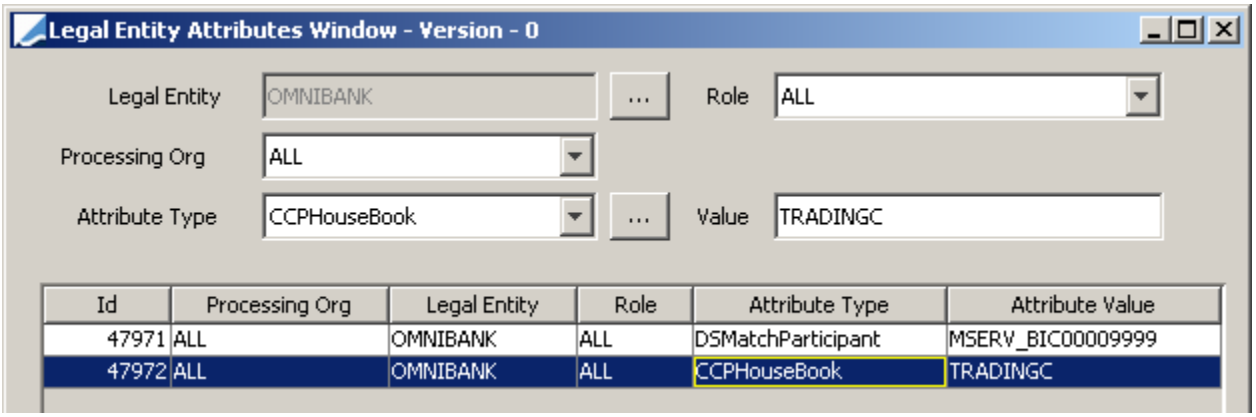
4.10 Legal Entity Mapping for Funds

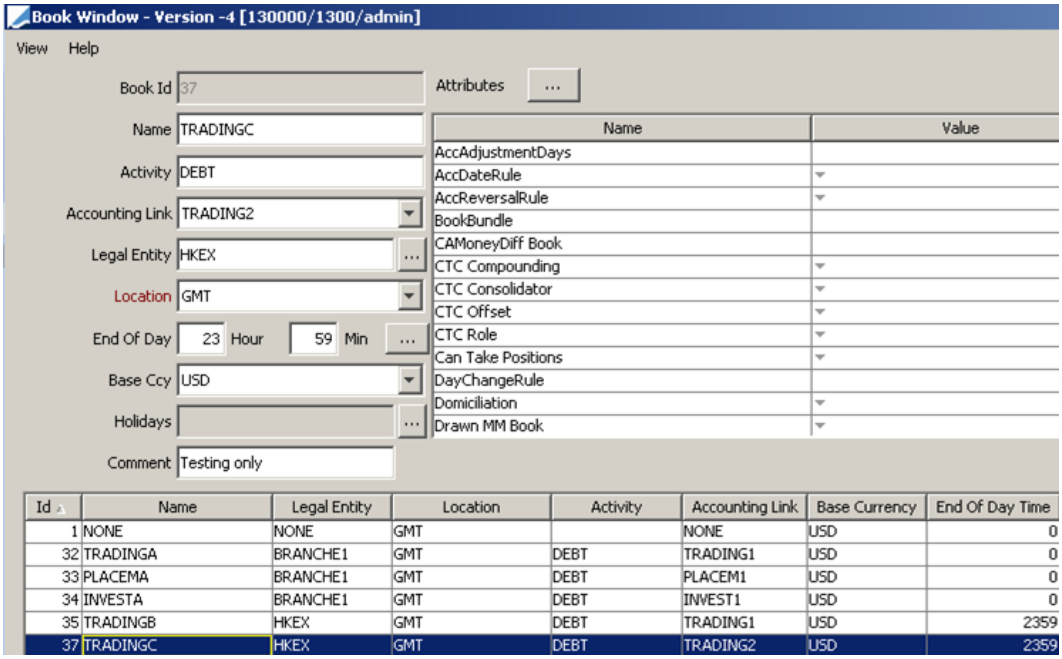


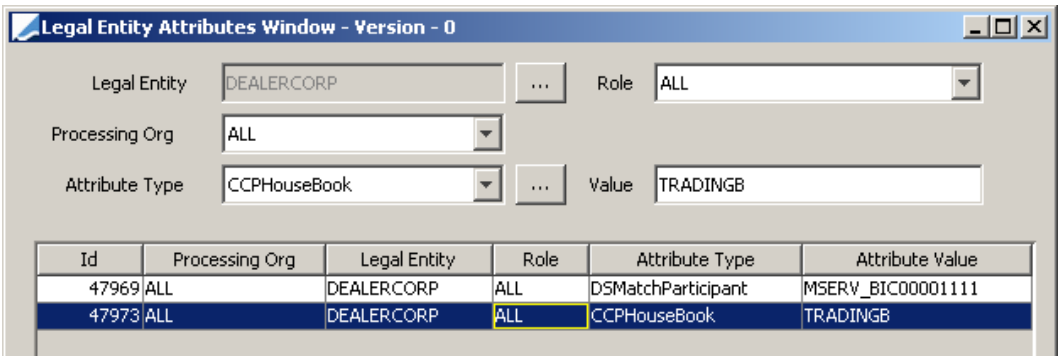
4.11 Book Mapping

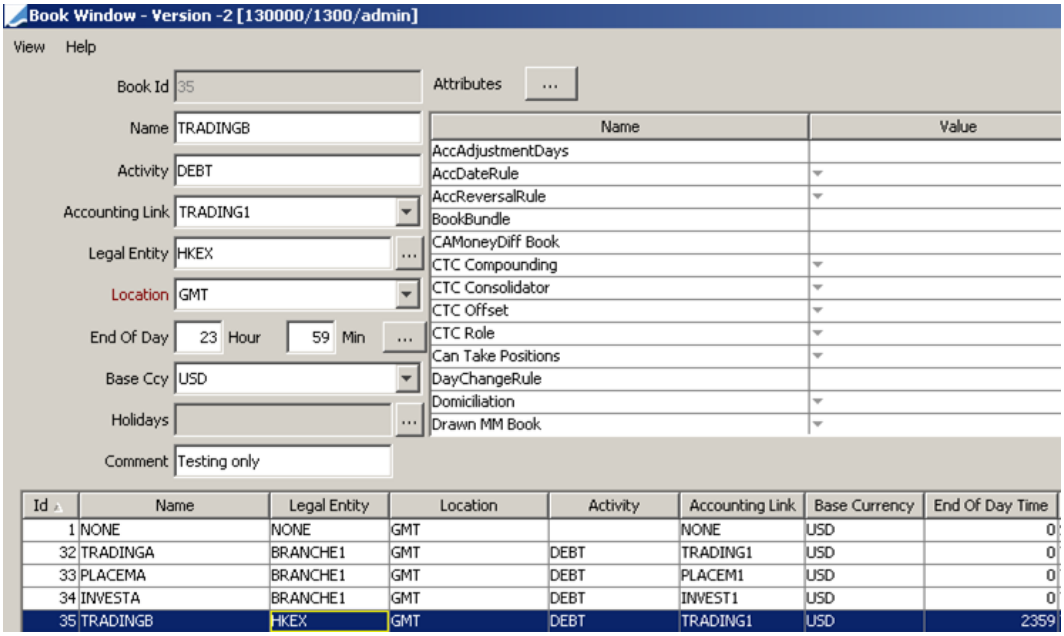
The FpML message is modeled as two trades. The CCP is the PO for both the trades. The Members are counterparty to the trades. Each member's trade is stored in their own books.

Once the legal entity (Counterparty) is identified as shown above, the book will be identified using the attribute CCPHouseBook on the chosen legal entity, the value of this attribute will give the Book to be used in Calypso and the PO of this book will be the CCP. For Client clearing the book is stored on the Broker legal entity with the attribute – CCPClientBook.


Legal Entity Attributes Window - Version - 0
Legal Entity
OMNIBANK
Role
ALL
Processing Org
ALL
Attribute Type
CCPHouseBook
Value
TRADINGC
Id
Processing Org
Legal Entity
Role
Attribute Type
Attribute Value
47971
ALL
OMNIBANK
ALL
DSMatchParticipant
MSERV_BIC00009999
47972
ALL
OMNIBANK
ALL
CCPHouseBook
TRADINGC


Book Window - Version -4 [130000/1300/admin]
View
Help
Book Id
37
Attributes
Name
TRADINGC
Activity
DEBT
Accounting Link
TRADING2
Legal Entity
HKEX
Location
GMT
End Of Day
23
Hour
59
Min
Base Ccy
USD
Holidays
Comment
Testing only
Name
Value
AccAdjustmentDays
AccDateRule
AccReversalRule
BookBundle
CAMoneyDiff Book
CTC Compounding
CTC Consolidator
CTC Offset
CTC Role
Can Take Positions
DayChangeRule
Domiciliation
Drawn MM Book
Id
Name
Legal Entity
Location
Activity
Accounting Link
Base Currency
End Of Day Time
1
NONE
NONE
GMT
NONE
USD
0
32
TRADINGA
BRANCHE1
GMT
DEBT
TRADING1
USD
0
33
PLACEMA
BRANCHE1
GMT
DEBT
PLACEM1
USD
0
34
INVESTA
BRANCHE1
GMT
DEBT
INVEST1
USD
0
35
TRADINGB
HKEX
GMT
DEBT
TRADING1
USD
2359
37
TRADINGC
HKEX
GMT
DEBT
TRADING2
USD
2359


Legal Entity Attributes Window - Version - 0
Legal Entity
DEALERCORP
Role
ALL
Processing Org
ALL
Attribute Type
CCPHouseBook
Value
TRADINGB
Id
Processing Org
Legal Entity
Role
Attribute Type
Attribute Value
47969
ALL
DEALERCORP
ALL
DSMatchParticipant
MSERV_BIC00001111
47973
ALL
DEALERCORP
ALL
CCPHouseBook
TRADINGB



4.12 Mapping for Acknowledgements

The following domain values needs to be mapped for sending RECEIVED/CLEARED/DECLEARED/REJECTED acknowledgements back to MARKITSERV.

Domain Values:

Clearing.Trade.RejectStatus

Clearing.Trade.RejectAction

DeClearing.Trade.RejectStatus

DeClearing.Trade.RejectAction

Clearing.Trade.AcceptStatus

Clearing.Trade.AcceptAction

DeClearing.Trade.AcceptStatus

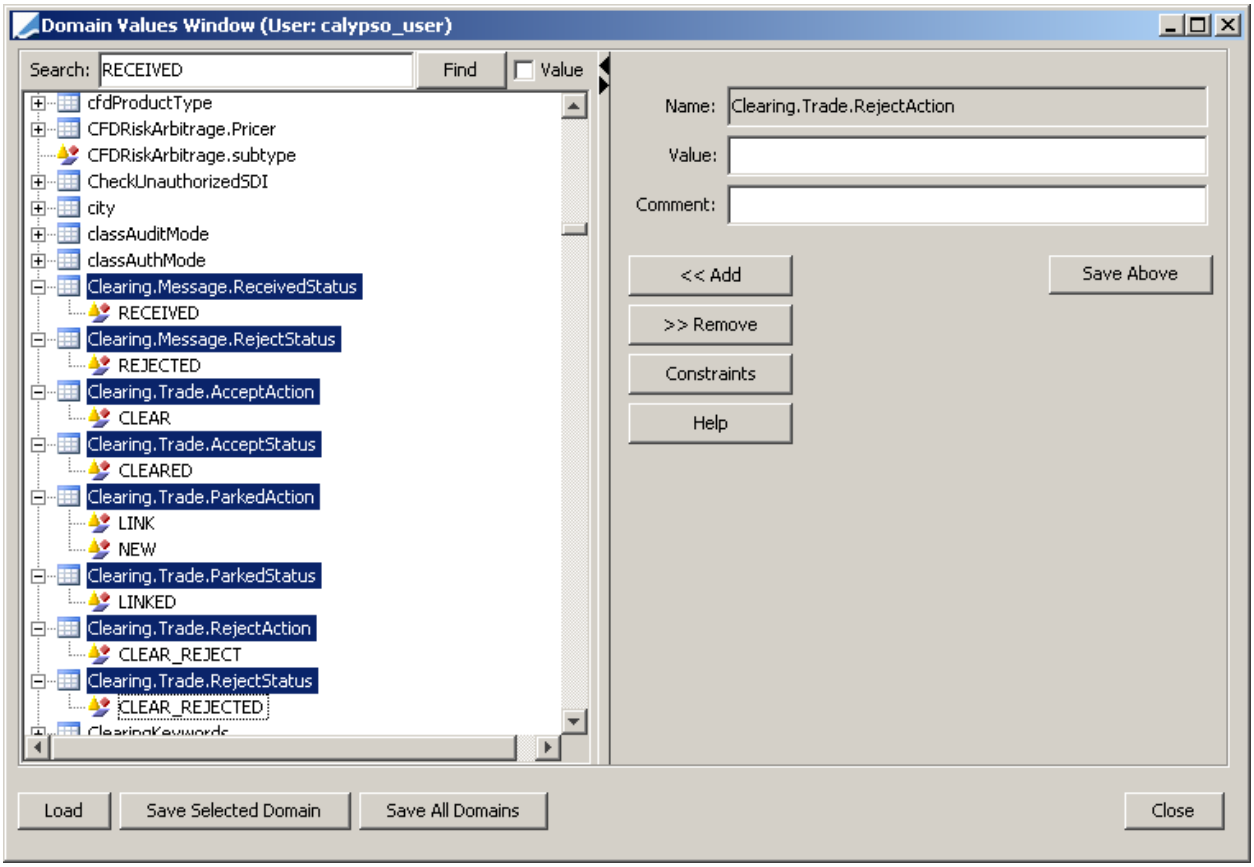
DeClearing.Trade.AcceptAction

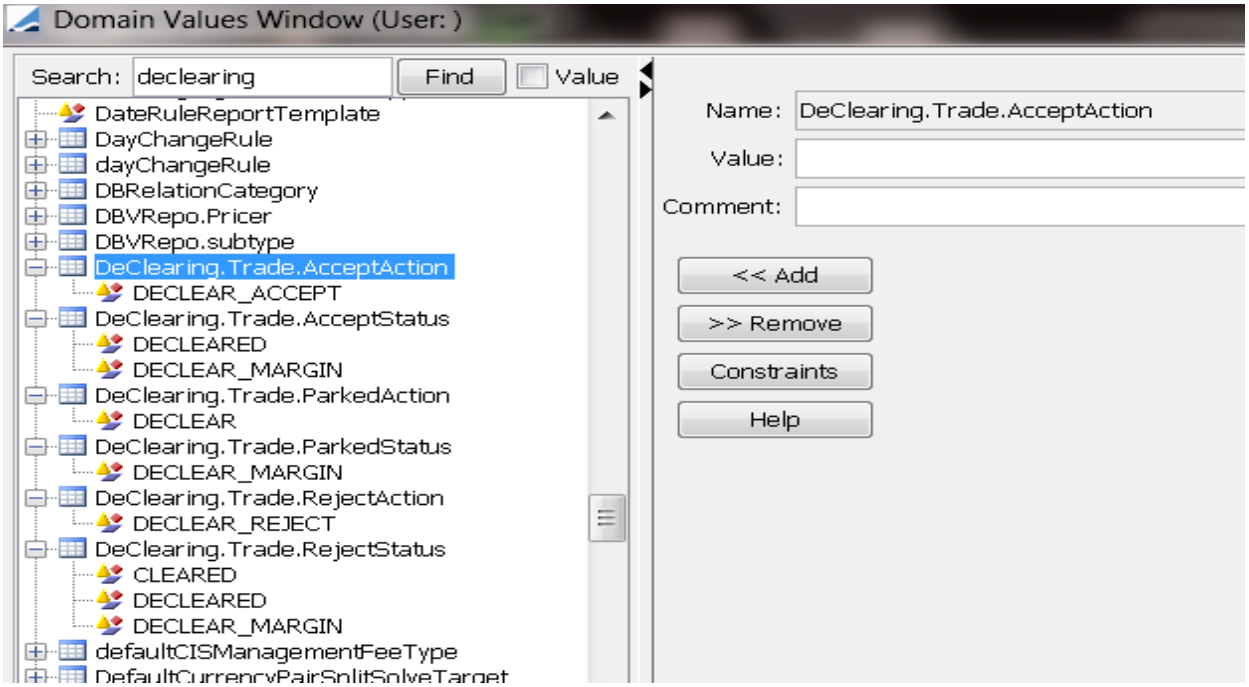
Clearing.Trade.ParkedStatus

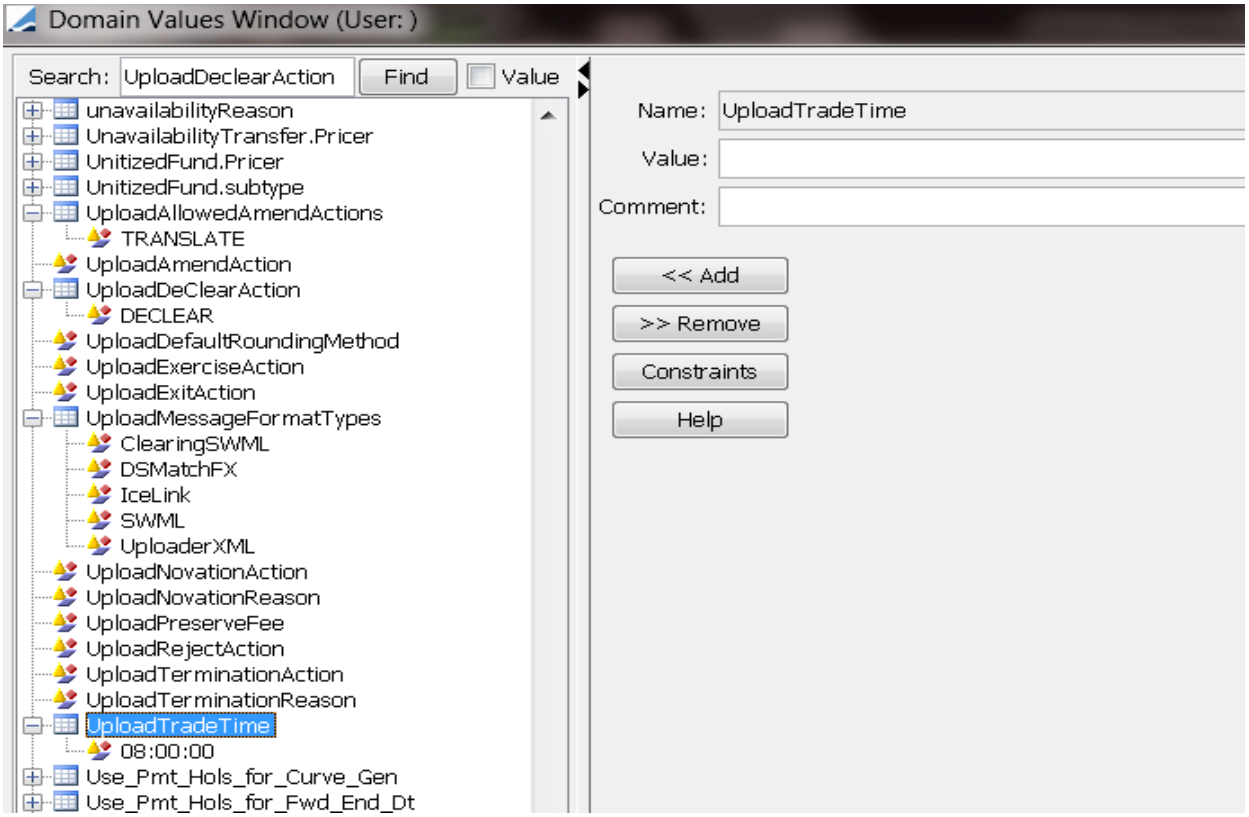
Clearing.Trade.ParkedAction

Clearing.Message.RejectStatus

Clearing.Message.ReceivedStatus


Domain Values Window (User: calypso_user)
Search: RECEIVED
Find
Value
cfdProductType
CFDRiskArbitrage.Pricer
CFDRiskArbitrage.subtype
CheckUnauthorizedSDI
city
classAuditMode
classAuthMode
Clearing.Message.ReceivedStatus
RECEIVED
Clearing.Message.RejectStatus
REJECTED
Clearing.Trade.AcceptAction
CLEAR
Clearing.Trade.AcceptStatus
CLEARED
Clearing.Trade.ParkedAction
LINK
NEW
Clearing.Trade.ParkedStatus
LINKED
Clearing.Trade.RejectAction
CLEAR_REJECT
Clearing.Trade.RejectStatus
CLEAR_REJECTED
Name: Clearing.Trade.RejectAction
Value:
Comment:
<< Add
>> Remove
Constraints
Help
Save Above
Load
Save Selected Domain
Save All Domains
Close


Domain Values Window (User:)
Search: declearing
Find
Value
DateRuleReportTemplate
DayChangeRule
dayChangeRule
DBRelationCategory
DBVRepo.Pricer
DBVRepo.subtype
DeClearing.Trade.AcceptAction
DECLEAR_ACCEPT
DeClearing.Trade.AcceptStatus
DECLEARED
DECLEAR_MARGIN
DeClearing.Trade.ParkedAction
DECLEAR
DeClearing.Trade.ParkedStatus
DECLEAR_MARGIN
DeClearing.Trade.RejectAction
DECLEAR_REJECT
DeClearing.Trade.RejectStatus
CLEARED
DECLEARED
DECLEAR_MARGIN
defaultCISManagementFeeType
Name: DeClearing.Trade.AcceptAction
Value:
Comment:
<< Add
>> Remove
Constraints
Help


Domain Values Window (User:)
Search: UploadDeclearAction
Find
Value
unavailabilityReason
UnavailabilityTransfer.Pricer
UnitizedFund.Pricer
UnitizedFund.subtype
UploadAllowedAmendActions
TRANSLATE
UploadAmendAction
UploadDeClearAction
DECLEAR
UploadDefaultRoundingMethod
UploadExerciseAction
UploadExitAction
UploadMessageFormatTypes
ClearingSWML
DSMatchFX
IceLink
SWML
UploaderXML
UploadNovationAction
UploadNovationReason
UploadPreserveFee
UploadRejectAction
UploadTerminationAction
UploadTerminationReason
UploadTradeTime
08:00:00
Use_Pmt_Hols_for_Curve_Gen
Use_Pmt_Hols_for_Fwd_End_Dt
Name: UploadTradeTime
Value:
Comment:
<< Add
>> Remove
Constraints
Help

MQ Setup

The DSMatch interface reads messages from a JMS Provider (like IBM MQ) via the DSMatch engine and translates them into CalypsoUploadDocument format. This is then passed to the Data Uploader for validation/upload into calypso.

The Following are the steps to be followed to support real time messaging via IBM MQ or any other JMS Provider like Apache Active MQ.

5.1 Pre Requisites

- Installation - IBM MQ / Active MQ (or another JMS Provider)
- Queues / Queue manager should already be created by customer
- Calypso with DSMatch & DataUploader configured.

5.2 DSMatch Configuration for JMS Support

This section covers installation and configuration of JMS Support in DSMatch.

5.2.1 Installation

Ensure that appropriate jars are set in the Classpath. Each JMS provider will have different jars required. We are providing an example for Websphere MQ and Active MQ below.

DSMatch

The dsmatch jar should already be in classpath.

For IBM WebShere MQ:

com.ibm.mq.commonservices.jar

com.ibm.mq.headers.jar

com.ibm.mq.jar

com.ibm.mq.jmqi.jar

com.ibm.mq.jms.Nojndi.jar

com.ibm.mq.pcf.jar

com.ibm.mqjms.jar

connector.jar

fscontext.jar

jms.jar

jndi.jar

providerutil.jar

com.ibm.mq.jmqi.jar

dhbcore.jar

For Active MQ

activemq-core-5.5.0.jar (or whatever version the customer is using)

kahadb-5.5.0.jar

5.2.2 Calypso Configuration

The following configuration needs to be done in Calypso

- DSMatch engine for Receive Messages (Receiver Queue)
- Acknowledgements: DSMatch engine to send ACK / NACK messages to Sender Queue.

The DSMatch engine listens to JMS Queue (Receiver Queue) based on the Message engine configuration. Any SOAP (having FPML embedded) message placed on the queue is read by the engine and then translated to CalypsoUploadDocument and finally Uploaded into Calypso. Acknowledgements generated are sent back to the Sender Queue using the same Engine.

5.2.3 Configuring DSMatch Engine

To configure the DS Match engine, please change the following in the calypso_uploader_config.properties.

1. Connection Details for the JMS

Please uncomment the appropriate block for Active MQ or Websphere MQ. The sample below shows the connection for Active MQ, where as Websphere MQ is commented. So, please keep one of the blocks below

```
# Connection Details for JMS
# Start Connection Details for ActiveMQ
jms.url=tcp://localhost:61616
jms.modetypeclass=org.apache.activemq.jndi.ActiveMQInitialContextFactory
jms.queue.connectionFactory=ConnectionFactory
# End Connection Details for ActiveMQ


# Start Connection Details for IBM Websphere MQ
```

```
# Please see documentation (appendix) on how to generate
# this bindings files
#jms.url=file://localhost/usr/local/calypso/resources/dsmatch/binding
#jms.modetypeclass=com.sun.jndi.fscontext.RefFSContextFactory
#jms.queue.connectionFactory=QueueConnectionFactory
# End Connection Details for IBM Websphere MQ
```

2. Input Queue Name. Please keep one of the blocks below based on the JMS provider.

```
# Input Queue Name (Active MQ)
# Note that for ActiveMQ the queue name has to prefix with
# 'dynamicQueues/', so the following is needed for Active MQ
input.queue.name=dynamicQueues/inputqueue
dynamicQueues/inputqueue.queue.setContext=true

# Input Queue Name (Websphere MQ)
input.queue.name=inputqueue
inputqueue.queue.setContext=true
```

3. Output Queue Name: Please keep one of the blocks below based on the JMS provider.

```
# Output Queue Name (Active MQ)
# Note that for ActiveMQ the queue name has to prefix with
# 'dynamicQueues/', so the following is needed for Active MQ
output.queue.name=dynamicQueues/outputqueue
dynamicQueues/outputqueue.queue.ackType=auto
dynamicQueues/outputqueue.queue.persist=true
dynamicQueues/outputqueue.queue.transacted=false

# Output Queue Name (Websphere MQ)
output.queue.name=outputqueue
outputqueue.queue.ackType=auto
outputqueue.queue.persist=true
outputqueue.queue.transacted=false
```

4. MQ user name

The below step is to be done for connecting to a remote queue manager having the user configuration done as previously mentioned.

```
# Set the following property having the username of the MQ user setup on remote queue manager.

jms.queue.connectionUserName=mq_user
```

5. MQ user password

```
# Set the following property having the password of the MQ user setup on remote queue manager.

jms.queue.connectionPassword=<password>
```

6. App Id Data user and password

```
# Set the following properties having the username and password given by DTCC which will be set as
ApplIdenttityData field in the message header.

<OutputQueueName>.queue.appiddatauser=<username>

<OutputQueueName>.queue.appiddatapass=<password>
```

5.2.4 Start the DSMatch Engine

The DSMatch engine can be started from the Engine Manager in Web Admin.

Appendix: Websphere MQ Setup

The following steps are to be performed only when using IBM MQ Series.

IBM WebSphere MQ Installation

1. Install IBM MQ Series.

2. Create Queues.

- Launch MQ Explorer and create the following
- Queue Manager
- Input Queue
- Output Queue

3. Create Binding File for JMS Connectivity.

Binding File for JMS Connectivity

Websphere MQ requires a bindings file to be created for JMS usage. This file has configuration which allows Java to connect to Websphere as a MQ. The process is as follows

- Generate Bindings file as explained below, which is used as PROVIDER_URL to connect to the Middleware.
- Link the Bindings file in the DSMatch Import Message Engine for JMS Provider setup. E.g. see the line below to be changes in the DSMatch Import Message Engine configuration file.

```
#jms.url=file://localhost/usr/local/calypso/resources/dsmatch/binding
```

6.1 Generating Bindings

- Backup your **JMSAdmin.config** file if already present in environment. Rename the file **src/main/resources/dsmatchJMSAdmin.config** to **JMSAdmin.config**

This is required since IBM tools to generate bindings only read JMSAdmin.config. So, please be sure to backup any earlier JMSAdmin.config file so that it can be restored.

- Edit the file **src/main/resources/JMSAdmin.config** and change the following properties

PROVIDER_URL=file:/usr/local/calypso/resources/dsmatch/binding (this indicates the path to the bindings file and the name of the file will be ".bindings";)

SECURITY_AUTHENTICATION=none (authentication model)

- Edit the file **src/main/bin/dsmatchMQSeries.bat** and change the highlighted text (which are the Queue Manager Name, Input queue name and Output queue name).

```
@echo off


rem ------------------------------------------------------------

rem  this batch file is used to generate the bindings that are used by the connection factory.

rem  replace InputQueueName with the name of the input queue

rem  replace OutputQueueName with the name of the output queue

rem  replace QM_Name with the name of the Queue Manager

rem  change the classpath to the appropriate path

rem ------------------------------------------------------------


echo + Creating script for object creation within JMSAdmin

echo del qcf(QueueConnectionFactory) > dsmatchmqsetup.scp

echo del q(CalypsoToClient) >> dsmatchmqsetup.scp

echo del q(ClientToCalypso) >> dsmatchmqsetup.scp

echo def qcf(QueueConnectionFactory) TRAN(CLIENT) HOST(127.0.0.1) PORT(1414) qmgr(QM.JM)
CHANNEL(SYSTEM.ADMIN.SVRCONN)>> dsmatchmqsetup.scp

echo def q(CalypsoToClient) qu(CalypsoToClient) qmgr(QM_CALYPSO) tc(JMS)>> dsmatchmqsetup.scp

echo def q(ClientToCalypso) qu(ClientToCalypso) qmgr(QM_CALYPSO) tc(JMS)>> dsmatchmqsetup.scp

echo end >> dsmatchmqsetup.scp


set CLASSPATH="D:\\calypso\\software\\rel12sp6\\mq-jars\\com.ibm.mq.jar;D:\\calypso\\software\\rel12sp6\\mq-
jars\\com.ibm.mqjms.jar"

echo + CLASSPATH=%CLASSPATH%

echo + Calling JMSAdmin in batch mode to create objects

java -DMQJMS_LOG_DIR="%MQ_JAVA_DATA_PATH%"\log -DMQJMS_TRACE_DIR="%MQ_JAVA_DATA_PATH%"\errors -
DMQJMS_INSTALL_PATH="%MQ_JAVA_INSTALL_PATH%" com.ibm.mq.jms.admin.JMSAdmin < dsmatchmqsetup.scp


echo + Administration done; tidying up files

del dsmatchmqsetup.scp
```

where

- QM_CALYPSO is Queue Manager
- CalypsoToClient is Input Queue
- ClientToCalypso is Output Queue

We are giving a sample for Windows. Please create a similar script for Unix and change to Unix format (% should change to $ for variable names, change CLASSPATH to use Unix syntax).

Sample script for linux environment:

```
#!/bin/sh

#This script is used to create a MQ binding file. Please make sure that the directory specified in
JMSAdmin.config PROVIDER_URL exists in the disk. The usages of the script is as below:

# sudo -u <user> ./createMQbinding.sh QM TEST.SENDQ TEST.RECVQ  <hostname> <port> <client_name>
<client_env_name>


QUEUE_MANAGER=$1

INPUT_QUEUE=$2

OUTPUT_QUEUE=$3

MQ_HOST=$4

MQ_PORT=$5

CLIENT=$6

ENV=$7


USER_NAME=$USER


#USER_NAME=who

BINARY_PATH="/usr/local/calypso/clients/${CLIENT}/${ENV}/jars"


SCP_FILE="${ENV}_`date +'%Y%m%d%H%M%S'`.scp"


echo "del q($INPUT_QUEUE)" >> $SCP_FILE

echo "del q($OUTPUT_QUEUE)" >> $SCP_FILE

echo "def qcf(QueueConnectionFactory) TRAN(CLIENT) HOST($MQ_HOST) PORT($MQ_PORT) qmgr($QUEUE_MANAGER)
CHANNEL(SYSTEM.ADMIN.SVRCONN)" >> $SCP_FILE

echo "def q($OUTPUT_QUEUE) qu($OUTPUT_QUEUE) qmgr($QUEUE_MANAGER) tc(JMS)" >> $SCP_FILE

echo "def q($INPUT_QUEUE) qu($INPUT_QUEUE) qmgr($QUEUE_MANAGER) tc(JMS)" >> $SCP_FILE

echo "end" >> $SCP_FILE


CLASSPATH="${BINARY_PATH}/com.ibm.mq.jar:${BINARY_PATH}/com.ibm.mqjms.jar"

echo "+ CLASSPATH=$CLASSPATH"

echo "+ Calling JMSAdmin in batch mode to create objects"

java -cp $CLASSPATH -DMQJMS_LOG_DIR="/home/$USER_NAME" -DMQJMS_TRACE_DIR="/home/$USER_NAME" -
DMQJMS_INSTALL_PATH="/home/$USER_NAME" com.ibm.mq.jms.admin.JMSAdmin < $SCP_FILE


echo "+ Administration done; tidying up files"

# rm $SCP_FILE
```

- This batch file or the shell scipt calls the IBM JMSAdmin tool that looks for the property file JMSAdmin.config in the current directory. So, please run the script in the same directory where the JMSAdmin.config is present. This will create the binding file.
- After generating the bindings you can set the DSMatch Import Message Engine configuration appropriately.
- Run the batch file dsmatchMQSeries.bat as shown below

```
C:\calypso\software\dsmatch\src\main\resources>C:\calypso\software\dsmatch\src\main\bin\dsmatchMQSeries.bat

+ Creating script for object creation within JMSAdmin

+ Calling JMSAdmin in batch mode to create objects

5724-H72, 5655-L82, 5724-L26 (c) Copyright IBM Corp. 2002,2005. All Rights Reserved.

Starting Websphere MQ classes for Java(tm) Message Service Administration

JNDI initialization failed, please check your JNDI settings and service.

For additional information on the cause of this problem run with the -v argument

+ Administration done; tidying up files

+ Done!
```

Support for Reporting Keywords

DSMatch sends the USI/UTI values in the incoming trade, and we add the same on the Calypso trade as trade keywords.

As part of the acknowledgement, we send out the CCP generated USI/UTI back to DSMatch.

7.1 Incoming Messages

We support the following keywords from the incoming messages for the CFTC (DoddFrank) jurisdiction and the HKMA jurisdiction respectively:

- ReportingCFTCUSIPrefix
- ReportingCFTCUSIValue
- ReportingHKMAUTIPrefix
- ReportingHKMAUTIValue

These are the Alpha or bilateral USI/UTI which CCP receives from the platform. Both are populated if received in the incoming message.

7.2 Outgoing Ack message

CCP generates a Cleared USI / UTI and performs the following:

The following is the process to be followed:

1. Incoming message comes from DSMatch for Clearing to the CCP for both sides.

2. We save the USI/UTI values on the Calypso trade as indicated above.

3. The CCP will generate the new USIPrefix/USIValue for each side. CCP will move the current values in ReportingHKMAUTIPrefix and ReportingHKMAUTIValue into the ReportingHKMAPriorUTIPrefix and ReportingHKMAPriorUTIValue respectively and ReportingCFTCUSIPrefix and ReportingCFTCUSIValue into ReportingCFTCPriorUSIPrefix and ReportingCFTCPriorUSIValue respectively.

4. The CCP will populate the ReportingHKMAUTIPrefix, ReportingHKMAUTIValue, ReportingCFTCUSIPrefix and ReportingCFTCUSIValue with the new values that are generated by the CCP.

5. CCP will clear the trade which will make the new values to be sent to DSMatch platform as part of the Clearing accepted acknowledgement. The acknowledgement will have keywords from both sides of the trades.

The picture below shows the list of keywords on Calypso trade:

ReportingCFTCPriorUSIPrefix	1030338961
ReportingCFTCPriorUSIValue	MSFX20130204000000000000000138381
ReportingCFTCUSIPrefix	1030338963
ReportingCFTCUSIValue	MSFX20130204000000000000000138383
ReportingHKMAPriorUTIPrefix	1030338962
ReportingHKMAPriorUTIValue	MSFX20130204000000000000000138382
ReportingHKMAUTIPrefix	1030338964
ReportingHKMAUTIValue	MSFX20130204000000000000000138384