CBOED*irect* Release Notes

CboeD*irect* 8.6

Last updated: 1/27/2011

Authors: Ravi Vazirani

Table of Contents

Overview of CD Server changes in this release

Work requests included in this release

- 6208 – SPX Linkage
- 6538 – CDX Fast Failover
- 6652 – CDX Light order API
- Xxxx – Infra 14
- 6834 - Removal of Market Data from Order History for Cxl Reports.
- 6826 – Order and Quote Trnsaction Optimization.
- 6648 – LT6 Phase 2 (New Cancel Report)
- 6758 – CBOE Execution Services

WR 6208 : SPX Linkage

Goal

The prime objective of the SPX Linkage Project is to enhance BOB functionality so that it supports the distributive Linkage plan (Allows SPX to be multi listed at C1 and C2). This project aims at converting manual quotes into orders and enabling the acceptance of multiple Manual quote orders for the same product at different price points and supports automatic trading against these manual quote orders. The project also aims at preventing auto execution against autoquotes.

Summary of changes

Following areas of the system have been changed to allow for SPX Linkage

- PAR – enter Manual Quote Order from PAR (origin code = 'K').
- PAR – New Fish button to sweep intrest in C2.
- OHS – Move SPX Origin/Contingency Table from XML in TE to OHS trading property.
- OHS – Acceptance of new Manual Quote Orders.
- TE – Changed Customer and Non-customer ISO/IOC processing in TE.
- TE – Changed customer and Non-customer non – ISO/IOC processing in TE.
- TE – Bob Classes can now Sweep C2 Markets (new Linkage for BoB)
- TA button for SPX Can SAL now.
- Auto Cancel Manual Quotes on User Logout, Slow failover and when the product goes into HALT state.
- CAS/FIX layer to disable drop copy publish to users for Manual quote orders.
- New allocation algorithms are created to give preference to Manual Quote Orders.

Note For more details on this project please refer to the detailed requirement document as that explains the new functionality being added in detail.

New Routing, Trading and Firm properties

1. Enable New BOB – *Firm Property*

We need to set this property only at the option level.

This property will allow an index Hybrid class to be enabled for "New BOB" processing.

This will be defined by class and should only be used for a class that is already marked as INDEX HYBRID (Setting this property on a non-Index hybrid class will cause undeterministic behavior)..

Setting this to "false" will enforce the index hybrid class to follow old BOB path.

A script will be run initially at install time to turn "New BOB" off.

2. New BOB Origin Code Contingency Type Mapping – *Routing Property*

We need to set this property at the option level & Spread level.

This property replaces the xml file that was used to allow certain orders to be routed to the trade engine. Based on the origin and contingency of the order, this property will route the order to TE, PAR, OMT or reject back to the user. This property will only allow valid origin codes to be input. The valid origins are B, F, I, X, M, N, and K. Please note that these are non-customer origins. The origins that are configured for customer will bypass this edit check and be routed to TE.

All non-customer bob orders will be checked against this table. The different outcomes are detailed below:

1.) If the Origin is Not Found in the table the order will reject back to the User.

2.) If the Origin is found in the Table and the Contingency of the order matches the allowed contingecies in the Table, the order will be routed to TE.

3.) If the Origin is found in the table, but the contingency of the incoming order is not found in the table, the order will follow existing workflow for orders where the contingency was not found in the table. Most likely ending up on firm/class/correspondent PAR.

3. Allowed SAL Origin Codes – *Trading Property*

We need to set this property at the option level and at spread product level.

This property is used to start SAL by the origin of the Order that was routed into CBOEDirect. For "New BOB", a SAL auction can be started for both customer and non-customer orders. Once SAL is turned on , this property will be used to further decide which origins are valid to start a SAL auction. If the origin is not enabled with this property, this order will NOT start a SAL auction.

A script will be run to set customer (origins c and w) on, and Non-customer off.

New Allocation Strategies for SPX

For BOB classes the following allocation stratagies will be used:

We are only changing two of them, regualar allocation and Hal allocation. We are adding 33, 34, and 35. Below is exactly how the allocation should be defined.

1.) For “Regular” Allocation

Regular(0)

Pro Rata(2)

Customer(4), Manual Quote(33), Best DPM Complex/UMA(13)

2.) For “HAL” Allocation

Hal Allocation(2)

Pro Rata(2)

Customer_pre_auction(34), Manual_Quote_Pre_Auction(35), Customer(4), Capped UMA(15)

How to turn on “New BOB” functionality

1. Verify the “New Bob Origin Code Contingency Table” routing property is set.
2. Verify the “Sal Origin Code table” is set.
3. Verify the allocation strategies are set as expected.
4. Set the Firm property “Enable New Bob” for the default firm to true.

How to turn off “New BOB” functionality

1. Set the firm property “Enable New BOB” for the default firm to false.

WR 6538: CDX Fast Failover

Goal

The goal of this project is to achieve continuous availability of the BC layer which ultimately is a step towards continuous availability of the CBOEDirect application. The goal is to have the BC Failover time be 1 second or less.

Summary of changes

The failover time improvement or continuous availability will be achieved by avoiding all kinds of I/O during failover & keeping all data needed in memory and by not going through an elongated opening cycle. This strategy is implemented by making the following changes:

Note With the usage of the distributed cache the memory requirement for OHServer has increased. The vm memory settings for OHServer will be changed to a minimum of 6 gig.

- Synchronize order updates between the master and slave OHS instances.
- Synchronize order updates between master and slave Trade Server instances (Quote like orders IOC, and express orders are not synched up at this time as they are cancelled on failover).
- Synchronize the state of the trading products between the master and slave Trade Server processes.
- Eliminate the need for database queries on slave to master transition of the BC
- Keep all event channels connected on the slave BC, thus eliminating the need to perform the event channel connection during the slave to master transition.
- Remove CORBA object offer exports from the slave to master transition.
- Keep all the internal caches updated in the slave mode thus eliminating the need to perform global queries during the slave to master transition.
- Quotes will not be synchronized between master and slave at this point in time.
- Notify the firms that a failover on certain classes have occurred and that they need to re-enter their quotes.
- New failover command line scripts and SHM GUI have been created to be able to execute the new fast failover procedures.
- At all layers all processes have been changed to use the new logging service.
- Automated DataIntegrity checks have been added on the Slave side, every so often the slave side checks its internal caches and compares the counts to the corresponding caches on the master side, if out of synch by a certain percentage / error factor alarms are logged (We monitor the count of products, orders, ors ids, user events & product update / add event to ensure integrity of the slave side).
- **For more details on this project please refer to the corresponding project charter (as it explains in detail the changes made to every process to achieve this failover functionality).**

RFQ to firms

On a failover, since quotes are not going to be synched up on the slave side, there will be a mechanism added to notify the firms that their quotes have been deleted from the system and they need to re-enter their quotes.

On a failover, the Trade Server automatically publishes a “server failure event” with an appropriate reason code upon transition from slave to master.

This event will then be forwarded by the CAS to the end user.

The new failover procedure

Note The current procedure still works as is in production, to execute new failover new commands have to be explcitly used.

During normal operation

1. During normal operation, the Orders in the OHS and the Order book will be kept in synch between the master and slave BC’s (This will occur in OHS and Trade Server processes).
2. During normal operation the ORS ID’s will be kept in synch on the master and slave OHServer processes.
3. During normal operation the product state information and other information like the last traded price will be kept in synch between the master and Slave Trade Servers.
4. During normal operation the slave Trade Server will keep the market data updated (i.e. the current market as a result of orders in the book and the botr/nbbo received from OPRA). However no market data events bookdepth, current market or last sale will be generated by the slave Trade Server.
5. During normal operation the master and slave processes will connect to all event channels and keep any changes related to product information and user information in synch.
6. During normal operation both the master and slave Status Server and Copp Bridge will receive all events. The slave processes will ignore the data that is being consumed.
7. During normal operation both master and Slave CoppBridge will be connected to copp.

Note There are 3 different options you can pick from to execute a new failover and 2 different ways of doing this (either through the command line or the SHM GUI). Below we document the command line way of executing the failover. The SHM way is self explanatory when you bring the SHM GUI.

Fast Failover Command line Synopsis

```
failover [-noAsk] [-skipMaster] [-processList <process1,process2,..,processN]

-noAsk               -Will not prompt for confirmation
-skipMaster          -Will not try to kill master processes
-processList         -To specify which processes to failover
-disableMaster       -If this option is specified the master
                      side processes will not be killed but
                      their POAs state will be changed to DISCARDING
Example:
Failover -noAsk –skipMaster -processList HybridTradeServer1,StatusServer
```

Note The failover command can be run either on master side or the slave side.

Failover Option 1 – Kill the master bc

On the slave side of the bc you want to failover, type the following command and press <ENTER> key.

```
failover

ABOUT TO FAILOVER THE FOLLOWING PROCESSES:
ProdBC02x1OHServerHybridperfbc2a ProdBC02x1HybridTradeServer1perfbc2a
ProdBC02x1HybridTradeServer2perfbc2a
ProdBC02x1HybridTradeServer3perfbc2a
ProdBC02x1HybridTradeServer4perfbc2a ProdBC02x1CoppBridgeperfbc2a
ProdBC02x1StatusServerperfbc2a
FROM perfbc2b TO perfbc2a

Would you like to proceed?(yes/no): yes

FAILOVER WAS SUCCESSFUL - FAILOVER TIME WAS 250 millis.
```

Failover Option 2 – Skip the master bc

Use this option when you don't want to do anything with the master processes because you cannot get to the master box's.

On the slave side of the bc you want to failover , type the following command and press <ENTER> key.

```
failover -skipMaster

ABOUT TO FAILOVER THE FOLLOWING PROCESSES:
ProdBC02x1OHServerHybridperfbc2a ProdBC02x1HybridTradeServer1perfbc2a
ProdBC02x1StatusServerperfbc2a ProdBC02x1CoppBridgeperfbc2a
FROM perfbc2b TO perfbc2a

Would you like to proceed?(yes/no): yes

FAILOVER WAS SUCCESSFUL - FAILOVER TIME WAS 250 millis.
```

Failover Option 3 – Disable the master bc

Use this option when you want to disable the master processes before failing over so you can debug what happend to the master side after the failover is complete (for example you may want to do a full GC on master side but you want to failover first).

On the slave side of the bc you want to failover , type the following command and press <ENTER> key.

```
failover -disableMaster

ABOUT TO FAILOVER THE FOLLOWING PROCESSES:
ProdBC02x1OHServerHybridperfbc2a ProdBC02x1HybridTradeServer1perfbc2a
ProdBC02x1StatusServerperfbc2a ProdBC02x1CoppBridgeperfbc2a
FROM perfbc2b TO perfbc2a

Would you like to proceed?(yes/no): yes

FAILOVER WAS SUCCESSFUL - FAILOVER TIME WAS 250 millis.
```

The following happens on a failover

1. As part of executing the failover command , the command will automatically put the POA of the slave processes in discarding state.
2. As Part of Failover the trade server will publish a serverFailure event indicating a failover has occurred (with an appropriate reason code). This event will be consumed by Cas's. Cas's will notify each connected user that they need to re-enter their quotes. The notification will basically indicate a class level RFQ.
3. The Trade Server will internally flag each trading product with an internal failover state. This internal state will be reset once a quote is received on that product.
4. The Trade Server will queue a timer to refresh the latest current market to opra (This needs to be done as there are only orders in the book and no quotes). The refreshes for products will be staggered. The values of stagger interval and the number of products to refresh is specified either in a configuration file.
5. When the timer pops for a particular product the current market will be refreshed and the new market will be sent to OPRA if and only if the internal flag is set.
6. As and when new orders are accepted, HAL will start if there are no quotes in the system and CBOE is not the NBBO. Which will eventually cause orders to link away if new quotes are not entered in a timely fashion.

Data Integrity checks & New Alarms for fast failover

Automated DataIntegrity checks have been added on the Slave side, every so often the slave side checks the counts of its internal caches and number of events received and compares those to the corresponding counts on the master side, if out of synch by a certain percentage / error factor alarms are logged (We monitor the count of products, orders, ors ids , user update events and product update events to ensure integrity of the slave side).

Note If these new alarms are being logged it means the master and slave side of the bc are not in synch.

Data Integrity Tolerance Parameters – Trade Servers

1. **Order Count**: Is configured with a tolerance of 2 times the average inbound order rate (i.e. rate of number of orders added/removed from the cache).
2. **Order update count** : Is configured with a tolerance of 2 times the average number of orders updated in the cache.
3. **Trading product count** : is configured to be an exact match or 0% of the number trading products in the local cache
4. **Trading product open count** : Is configured to be an exact match or 0% of the number of the number of open trading products in the local cache.
5. **Trading product update count** : Is configured to be an exact match or 0% of the number of trading products updates.
6. **Trade session update count** : is configured to be an exact match or 0% of the number of trading session updates

7. **User count** : is configured to be an exact match or 0% of the number of users cached.
8. **User update count:** is configured to be an exact match or 0% of the number of users updated
9. **Trading Product State Count:** is configured to be exact match of products in open state.

Data Integrity Tolerance Parameters – OHS

1. **Order count :** is configured with a tolerance of 2 times the average number of orders added/removed from the cache.
2. **Order update count :** is configured with a tolerance of 2 times the average number of orders updated in the cache.
3. **Orsid count** : is configured to be an exact match or 0% of the master side number of free orsids count.
4. **Orsid index count** : is configured to be an exact match or 0% of the master side orsid index count.
5. **Product count** : is configured to be an exact match or 0% of the number trading products cached.
6. **Product update count:** is configured to be an exact match or 0% of the number of trading products updates
7. **User count** : is configured to be an exact match or 0% of the number of users cached.
8. **User update count** :is configured to be an exact match or 0% of the number of users updated

Example of alarms

<u>OHS example – when the product event count is different between master and slave</u>

OHServerHybrid.log:high systemAlarm 2010/09/13 13:46:50.045 ProdBC02x1OHServerHybridperfbc2a BC02x1 "DataIntegrityServiceHome:DataIntegrityServiceOhsImpl>>> DISHI_SYNCHPOINTCOMPARE_DISCREPANCY - Master and Slave synchpoints have counts that are out of range"
high systemAlarm 2010/09/13 13:46:50.045 Generic ProdBC02x1OHServerHybridperfbc2a BC02x1 "DataIntegrityServiceHome:DataIntegrityServiceOhsImpl>>> Count[**productEvent**] ComparisonResult=[ABOVE RANGE] MasterCount=[1415] Slave Count=[1416]" 2

<u>TE example – when the Trading product count is different between master and slave</u>

high systemAlarm 2010/09/15 15:14:23.413 Generic test15HybridTradeServer2devenv1 "DataIntegrityServiceHome:DataIntegrityServiceTeImpl>>> DISHI_SYNCHPOINTCOMPARE_DISCREPANCY - Master and Slave synchpoints have counts that are out of range" 1
high systemAlarm 2010/09/15 15:14:23.413 Generic test15HybridTradeServer2devenv1 "DataIntegrityServiceHome:DataIntegrityServiceTeImpl>>> Count[**tradingproduct**] ComparisonResult=[ABOVE RANGE] MasterCount=[392] SlaveCount=[784]" 2
high systemAlarm 2010/09/15 15:14:23.413 Generic test15HybridTradeServer2devenv1 "DataIntegrityServiceHome:DataIntegrityServiceTeImpl>>> Count[user] ComparisonResult=[ABOVE RANGE] MasterCount=[26] SlaveCount=[52]" 3

<u>When ORS Ids are not in synch at startup on the slave side</u>

High systemAlarm ORsIDGeneratorImpl:ORSID Free pool is not available, FAST Failover is not possible for this BC.

WR: 6652 CDX Light Order API

Goal

Goal of the project is to provide a light and fast order entry interface that customers can use in lieu of quotes to take the added advantages of tiered quoting as well as one sided quoting.

Summary of Requirements

1. Light Orders are supported for both options as well as stocks.
2. Cancel Requests for light orders are supported.
3. Market Orders are not allowed.
4. Light Orders are always treated as *DAY* orders.
5. The only contingency that is allowed is *IOC*.
6. Light Orders are not allowed on BOB classes.
7. Light Orders are not allowed on restricted series.
8. Reasonability checks will be performed on the inbound Light Orders.
9. Light Orders can be canceled only through the cancel interface of Light Order API. Attempt to cancel a Light Order through the existing cancel interface will be rejected.
10. Light Order Cancel interface may be used to cancel only the Light Orders. Cancels targeting other orders will be rejected.
11. All Light Orders for a user will be canceled on logout.
12. Light Orders are not auction eligible. Also, Light Orders cannot be used to respond to auctions. However, Light Orders can possibly end an auction just like regular orders.
13. Light Orders are NBBO protected only if the NBBO protected flag is set to true in the incoming order.
14. Light Orders are never linked away. The order will be canceled if another exchange is at a better price and the NBBO protection is ON.
15. Light orders must be included in the opening and as part of the opening trades. If any remaining light orders would normally go through HALO, they still should as long as they are marked as NBBO protected. Light Orders that are not filled in HALO must not be routed away; instead they must be cancelled. Light Orders those are not marked as NBBO protected must be cancelled rather than go through HALO.
16. Light Orders will be subjected to a new rate limit that will be added as part of this project.

Configuration and Help Desk related requirements

1. Users need to be explicitly enabled to send Light Orders. Light Orders are enabled at a user acronym level.
2. There is a capability to set up a Light Order profile for a user through SA GUI. The profile includes the following information.
   a) Account -- *optional*
   b) subAccount -- *optional*
   c) executingGiveupFirm
3. Session level default values will be used unless there is class level overrides.
4. OMT single order query using the order ID (high/low ID or branch/ sequence) is supported. This query must be available for all OMT user types. The Light Order returned to the OMT will be View only. No actions can be performed on this order from OMT.
5. HDE order cancel functionality is supported for Light Orders.
6. SA GUI Order query is supported for Light Orders, with the following query types:
   a) Query by user and class
   b) Query by user and product
   c) Query by order ID (high/low ID or branch/sequence)
7. Each user must be configured with correct rate limits for each session. Default rate limits are respected but are set too low for some sessions which would not allow user to place any order/quote.

Order status reporting requirements:

1. A New Light Order request will not result in a NEW report in the case of CMI users. A normal return should be treated as success. FIX will send out a NEW unless some fill or cancel occurred while processing the order. When a fill or cancel occurs, FIX will send out only the execution report for that fill/cancel.
2. A New Light Order request with IOC contingency will not result in a separate Cancel Report if canceled.
3. A New Light Order that gets filled on entry will always result in a FILL report being sent out. This is to supply the contra party information that will not be available in the return struct.
4. A Light Order Cancel request will not generate a separate Cancel Report. A normal return should be treated as success. Return struct will provide the necessary information.
5. Any system cancel (other than logout, such as QRM if it ever happens) or a cancel through the Help Desk will result in a Cancel report being sent out on the External Order Status Channel. These Cancel reports are not published to the Status server and hence should not be ACKed.

API requirements

1. NEW Light Order request takes the following information:
   - ii. Branch
   - iii. Branch Sequence number
   - iv. productKey
   - v. Price
   - vi. Quantity
   - vii. Side
   - viii. userAssignedId
   - ix. positionEffect
   - x. Coverage
   - xi. Flag indicating nbboProtected or not
   - xii. Flag indicating IOC or not
   - xiii. session
   - xiv. originCode
   - xv. Cmta -- optional
   - xvi. Pdpm -- optional

2. A NEW Light Order response provides the following information:
   - i. highID and lowID
   - ii. Filled quantity – quantity that got filled while processing this inbound order.
   - iii. Canceled quantity – quantity that got canceled while processing this inbound order.
   - iv. leaves quantity – quantity that got booked for this order
   - v. retCode – return status code indicating the success or error if any occurred during the process.

3. A Cancel Light Order request must be able to take the following details:
   - i. branch and sequence number as supplied in the original order OR the Cboe high and low Id of the order.
   - ii. productKey
   - iii. UserAssignedCancelId
   - iv. session

4. A Cancel Light Order response provides the following information:
   - i. canceledQuantity – *quantity canceled*
   - ii. tradedQuantity – total quantity traded
   - *iii.* retCode – return status code.

Limitations/Restrictions with Light Orders

1. Cancel Replace of Light Orders are not supported.
2. No user query methods are supported for Light Orders.
3. Partial cancels are not supported.
4. The correspondentFirm will not be available in the case of Light Orders as it will be filled in with the userID internally. Moreover, because of this, the userIDs used for doing Light Orders may not be longer than 4 characters.
5. Origin codes that correspond to quote-like orders (e.g. “I”) are not allowed for Light Orders.
6. Bust Reinstate of Light Orders is not supported.
7. UserAssignedId and UserAssignedCancelId fields are limited to 8 characters

WR#6826 – Order and Quote Transaction Optimization

Summary

As part of overall efforts to improve user response time, the cost of the current technology used to support in-memory transactions has been re-evaluated. It has been observed via profiling and performance testing that the cost of the JGrinder reflection-based field-level transactional change control is responsible for up to one third of the time taken to process an order, order cancel, or quote operation. Of the business objects requiring in-memory transaction control, OrderImpl and QuoteImpl are modified with the highest frequency, when processing order/quote flow.

This objective of this project is to replace the field-level transactional change control with a custom-built transactional stack, which integrates with the JGrinder transaction boundaries but which does not incur the same costs as a typical transactional object.

JGrinder In-Memory Transaction Overview

The current transactional framework used for processing changes to business objects is a third-party framework called JGrinder. A business objects which extends a specific base class an implements various structural code for transactional fields achieves transparent transactional support.

When in the context of a transaction, a call to a setter does not modify the value of a field directly. Instead, an object is generated representing the previous and new values of the field. This object is added to a transaction log, which is managed by the JGrinder code base. A getter called by the same thread (within the context of the transaction) will be returned the value passed via the setter, while a getter called by another thread will still be returned the current field value. Transactions may be nested, meaning that there may be a stack of transaction logs.

When a transaction is committed, the changes in the transaction log are migrated to the outer transaction (if in a nested context), or are committed to the business object's fields (if a root transaction). Additionally, a root transaction may cause the changes to be committed to a persistent data store (most commonly a relational database). The trade server code base no longer has any direct database interaction on the order path, however, so this persistence feature is not relevant for the processing of quotes or orders.

There is additional support, via the CBOE Foundation Framework abstraction layer, for transaction listeners to be notified of root commit and rollback events, as well as inner transaction rollbacks.

Modified Transaction Implementation

Since the trade server usage of JGrinder is relatively simplistic, we can implement a modified transaction management implementation for the highest-frequency objects. All processing and reading of order/quote objects is executed within a separate lock, so there is no need for read-isolated transactions. Additionally, there is no database persistence of these domain objects from the trade server, so persistence support is also not a requirement. Nested transactions and partial rollbacks are, however, still required by the business processing logic.

Rollback TX Design

The change to the OrderImpl, QuoteImpl or OrderHandlingInstructions transactional support is to introduce an 'undo' change log for the transactional fields of these domain objects, with fields defined which are highly coupled to the respective domain object implementation itself. A bitmask is also used to signify if a particular field has a change to potentially 'undo' on a rollback. The 'undo' context object is associated with a TransactionLog, and is registered with the FoundationFramework's transaction listener mechanism. On a root commit callback, all undo logs are disposed of. On any rollback, the undo log is applied, replacing the current field values in the domain object implementation with their 'old' values (for any field marked as dirty in the undo log). Otherwise, there are no additional operations on the OrderImpl, QuoteImpl or OrderHandlingInstructions, as compared to a simple, non-transactional object.

When a setXxx() method is invoked, the change log entry for the domain object is found (or lazily created) for the current JGrinder TransactionLog. The value is marked as dirty in the bitmap, and the current value of the field is recorded for 'undo' operations. The setter then sets the new value directly on the domain object. If there is no current transaction, then the local field is simply set directly. This differs from the JGrinder setter implementation wherein the set call will either create a modified object record in the current transaction log, to be processed by all commit/rollback calls in the stack, and ultimately set back to the domain object via a reflective set call.

When a getXxx() method is invoked, the current value of the field is returned (i.e., the getter is implemented as a simple POJO getter, with no awareness of transactions). This is significantly different than the current JGrinder implementation, where getters must first check to see if they have a dirty value set in the transaction log.

Caveats to the new Transaction Implementation

Since the new transactional framework is loosely coupled to the JGrinder framework, it is unaware of all potential transaction boundaries, such as those in which there were no changes made to the domain object. In these cases, rollback code is complicated by the task of searching for the appropriate changes to undo, which incurs a $O(n^2)$ complexity. This is only an issue for transaction rollbacks, since commit calls have no cost, and is only an issue for transactions that are at least three levels deep.

The only other behavioral caveat, already noted in previous sections, is that this implementation does not provide for read-isolation in the transactions. This is not a concern in the trade server however, since the class (or product) lock will protect the order/quote from unsafe read operations.

OrderHandlingInstruction is a shared object between OHS and Tradeserver. In order to keep the changes localized to the trade server, a new implementation of HandlingInstructions (OrderHandlingInstructionNoReflectionImpl) is created.

WR: 6648 New Cancel Report (LT6-2)

Goal

The goal of this project is to eliminate latency in the Order Cancel Report by generating cancel reports in CAS layer. Current production Cancel Report is published by the Trade Server to the CAS.

Summary of changes

In this project, when a cancel request is sent by the user, the OHS and Trade Server will return a status back to the CAS in the same synchronous call that sent the cancel request. If the status is true it means the whole quantity of the order is fully cancelled, in this case the CAS will generate the cancel report for the user. If the status is false then it means the order was not cancelled in full and I this case the Trade Server will continue to generate the cancel report.

Behavior of New Cancel Report:

The Cancel Report will contain:

1. A Sequence of groups: each group contains a groupNameKey which is obtained from the PCS. The status server to filter products, and by the BC to ensure that it does not double publish uses this groupNameKey.
   In CAS layer, this group is currently get empty out in OrderQueryCache.java. Since the CAS does not use this group.

   This project, the cancel report that is generated by CAS will have empty value in group.

2. Order struct: Server send an orderStruct which contains:

   - Simple Order – Every Field remains the same as the original order except
     - `orderStruct.cancelledQuantity = orderStruct.originalQuantity;`
     - `orderStruct.leavesQuantity = 0;`
     - `orderStruct.state = StatusUpdateReasons.CANCEL; //2`

   - Strategy Order – Every Field remains the same as the original order except
     - `orderStruct.cancelledQuantity = orderStruct.originalQuantity;`
     - `orderStruct.leavesQuantity = 0;`
     - `orderStruct.state = StatusUpdateReasons.CANCEL; //2`
     - Leg - Every Field remains the same as original order except
       - Leg's Cancel Report – the Order Struct for the main package on the Cancel will look similar to the Order. Each leg will carry the cancel information for that particular leg and the preceding legs. **Example:** Cancel Report of Leg 1 will have Main package cancel quantity equal to original order and leg detail cancel quantity equal to leg one's detail original quantity. Other leg detail will remain the same as original

order. Cancel Report of Leg 2 will have all above info. In addition, Leg 2 detail cancelled quantity is equal to Leg 2 detail original quantity. The remaining leg detail will be the same as original order. Cancel Report of Leg 3 will have all above info. Leg 3 detail cancelled quantity is equal to Leg 2 detail original quantity. The remaining leg detail will be the same as original order. Cancel Report of Leg 4 will have all above info. Leg 4 detail cancelled quantity is equal to Leg 4 detail original quantity.

- AIM Order – Usually an AIM order will get traded or canceled. Users will only have a rare chance of cancelling this order. This feature will not likely apply to orders of this type. The only way to test this for AIM would be to extend the duration of the auction for longer time period, because the order is only available to cancel after an auction has ended.
- Cross Order – This will behave like an AIM order, except this is a stock order, so it will get traded or canceled. Users will only have a rare chance to cancel this order. This feature will not likely apply for this type of order. The only way to test this is to extend the auction for longer periods of time, because the order is only available to cancel after the auction has ended.
- Complex AIM Order – same as AIM

3. A sequence of Cancel Report structs: The CAS will fill out all the information in the cancel report struct, in the same manner as the server does except the ORDERID will not be available with a CAS generated cancel report.
4. Status change: Status will be NEW = 7, which is the same as the one that the server sends out in the current production version.

WR: 6834 Order History Optimization

Goal

The goal of the initial phase of this project is the removal of market data for cancel requests and responses from being appended to the Order History. The cost of this data gathering, conversion to structs, and publishing to event channels has been measured to be minimally 200 microseconds (µs), and up to 400µs. Business users have confirmed that there is no value in this historical data, therefore the Market Data information will be eliminated from the order history data that's generated by the trade servers.
Order history will no longer contain BOTR, NBBO, as well as linkage information for other exchanges.

Summary of changes

The Hybrid/C2 server generating market data for cancel reports, and sends the data back to OHS through the event channel. The struct for the market data on the publishCancelReport event will contain an empty struct quote query struct for the MarketData.
Both OHS and the TradeServer required changes.

WR 6758 : CBOE Execution Services

Goal

The goal of this project is to enhance the CBOEDirect system to support routing of linkage orders directly to other exchanges so CBOE can act as a Broker-Dealer.

Summary of changes

The LC code has been modified to support CXS as a special router.

If an outgoing Linkage order has been designated to route through CXS then the LC code will automatically routes directly to the desination exchange (in this release direct routing is provided only for CBOE and CBOE2 exchanges).

New dispatcher for CXS was added in the FixLinkageAdapter.

New formatter was added for CBOE and CBOE 2 exchanges (The formatter is used to format internal CBOEDirect messages for orders and fills into Fix messages that are sent to CBOE and C2).

A new fix engine will be configured on C1 and C2 to accept these linkage orders.

ugfixes included in this release

Remedy PR #	(PITS) Ticket ID	Assigned To	Create Date	Problem Description
SYS007585	13407	misbahud	5/13/2009 14:40	ORDERS SENT TO THE PAR SHOULD NEVER AUTO ANYTHING BUT ISTEAD BE MANUALLY HA
SYS008701	22194	mahart	3/1/2010 9:46	SPREAD TRADES ARE NOT REPORTED VIA THE CMI BUT THEY ARE SENT TO CLEARING. S
SYS008922	34015	manchana	6/2/2010 13:44	THE USER COULD NOT SEE THIS ORDER ON HIS OMT 9PTR1
SYS008923	34756	dowat	6/2/2010 13:53	A RACE CONDITION OCCURED ON A CANCEL/REPLACE ORDER.
SYS009025	39320	eckb	6/30/2010 10:08	OHS IS STILL ATTEMPTING TO GENERATE ETNS WHICH EVENTUALLY KICK-OUT TO OMT
SYS009078	40229	misbahud	7/27/2010 13:42	START AIM AUCTION ON CROSSES & LOCKS
SYS009087	42061	misbahud	7/28/2010 14:48	SERIOUS ORDER TRADES IN OHS AND PAR, DOUBLE FILL.
SYS009094	42944	gillundb	8/3/2010 9:38	ACR IN ORDER HIST FOR LINKAGE ORDERS IS USING XLK/ZLK INSTEAD OF REAL ACRON
SYS009114	42685	vazirani	8/11/2010 9:26	MARKET DATA ISSUE IN CBSX
SYS009119	43986	eckb	8/13/2010 8:42	WHEN A USER LOGIN GETS DISABLED THE ACRONYM IS GETTING DISABLED.
SYS009162	43107	mahart	8/25/2010 8:21	WHY DID THIS ORDER GO TO AUCTION WHEN BOTH INSTRUMENTS WERE NOT THE SAME?
SYS009265	46843	eckb	9/21/2010 16:41	MMHH -HAVE MMTN CARDED TRADES OUT OF SEQUENCE
SYS009291	47170	mahart	10/1/2010 10:26	MORGAN STANLEY HAS NOTICED SEVERAL ORDER ACK DELAYS
SYS009295	51180	eckb	10/4/2010 8:42	RE-COA DOES NOT HAPPEN IN FAST MKT STATE FOR REGULAR OR CPS COMPLEX ORDERS
SYS009308	49418	zhangj	10/6/2010 10:25	7 HK ORDERS AND ONE ADBE ORDER FILLED AT A WORSE PRICE.
SYS009316	51962	eckb	10/12/2010 11:18	1ST QUOTE WITH INVALID PRODUCT KEY

Remedy PR #	(PITS) Ticket ID	Assigned To	Create Date	Problem Description
SYS009324	51751	dowat	10/15/2010 8:53	IT LOOKS LIKE OHS IS NOT CREATING THE DSM AFTER THE INITIAL ROUTE FAILED A
SYS009338	50485		10/19/2010 15:30	ISSUE WITH DISABLING AN ACRONYM FROM MEMBERSHIP POSTPONED UNTIL 8.6 GOES I
SYS009432	57502	misbahud	11/17/2010 13:42	LINKAGE ISSUE WITH LKO CAUSING DUPLICATE ORDERS IDS BETWEEN C1 AND C2.
SYS009352	52764	mageem	10/12/2010 15:35	FIX MASSQUOTE ALLOWS QUOTES FROM MULTIPLE CLASSES
SYS009540	61718	mageem	12/22/2010 17:59	FIX ONE-STEP STRATEGY MESSAGES CAN THROW EXCEPTIONS IN PARSING LEG DATA THAT PROVIDE NO USEFUL FEEDBACK TO USER

Operator Procedure changes

1. As and when BC's are rolled out there should be no need to run the command businessExternalServices start any more , this is done automatically at startup and during a failover. **NOTE: It is harmless to run this command as if the connections are already started the command will not start them again.**
2. There are new commands to run the fast failover , these the documented in the section where Fast Failover project is described in this document.
3. Also new SHM GUI functionality has been added to execute Fast Failover.

Infra 14 Changes

Summary of changes

- *Infinispan & JCache*: The open-source Infinispan Cache product has been fully integrated into the Infrastructure in general and the Foundation Framework in particular, to provide a highly scalable data grid platform. This product is tied tightly to the CDX Fast Failover project.
- *Migrate Persistence Broker:* this new broker eliminates the “empty commit” issue with the Null Broker during de-queue, thus helps in persisting to a database rather than in memory.
- *“Giver”, or the Java Concurrent Actor System*: This system provides a new threading toolkit that implements the Actor design pattern and also provides a mechanism for attaching thread strategies to pipeline tasks. This concurrency project with the following 3 goals:
  1) Easy to use
  For instance, the actors have been designed to preserve normal-looking call semantics such as actor.send(message) and no massive framework is required. Things like publish/subscribe are built on top of the system so they can be easily changed. The implementation is less of a framework, and more of a collection of useful classes.
  2) Make the threading strategy configurable
  Give’r allows you to define a threading strategy for each actor. A few implementations have been provided, such as: Full Concurrency, FIFO, Inline, Thread-per-Actor, and AWTSwing.
  3) Make performance monitoring built-in
  Give’r includes the ability to monitor the latency of messages, and the time actors spend acting. An implementation is provided that monitors all messages and actors annotated with @Instrumented. The user may provide custom monitoring if desired.
- *Enhancements:*: The base code of the open-source Infinispan Cache product was enhanced to provide all features required by the CDX Fast Failover project team.
- *Maintenance*: Several code “clean-up” tasks were completed to eliminate obsolete and/or deprecated code. The DN code was refactored for ease of maintenance & availability. Other maintenance tasks were done to eliminate known errors.
- *Latency*: The Grizzly platform was enhance to utilize “blocking” reads on dedicated threads, as opposed to the Selector model with a thread hand-off

Branch Manifest

Branch Name	**Description**
infra_13.0_bomba_CDX_Fast_Failover_POC_Placeholder	Placeholder while doing CDX Fast Failover work with the Server team in case we want to keep any modifications.
infra_13.1_kothuri_MigratePersistenceBroker	MigratePersistenceBroker
infra_13.1_kothuri_jgroup_mcast_xml_generation	script template for generating jgroups multicast files to be used by server processes
infra_13.1_kothuri_POA_StateMgmt_For_FastFailover	POA State management for fast failover: DISCARDING to HOLDING for ALL UserPOA's
infra_13.1_eccles_dn_refactor_phase5_5_journal	Quick migration of the journal code to common and adding it to AMQ
infra_13.1_jia_genxtpExtentMap	generate XTP Extent Maps with perl scripts; move away from the template
infra_13.1_jia_registertoPW	register infra process with Process Watcher during each process' startup phase
infra_13.1_jia_cleanupDeprecatedES	clean up the deprecated methods in the Foundation Framework Event Service packages
infra_13.2_jia_cleanupDEC	clean up deprecated functions and rewrite some others in all Infrastructure VOBs
infra_13.2_bomba_cdx_add_jcache_to_ff_coreservice	add the JCache implementation as a Foundation Framework "core" service selection
infra_13.2_kothuri_jgroupschanges_installscripts	adding the jgroups changes to the install scripts
infra_13.2_kothuri_common_give_r	give-r: Java Concurrent Actor System
infra_13.2_bomba_cdx_jcache_fixes	Any last additions and/or bug fixes from incorporating JCache and Infinispan
infra_13.2_bomba_cdx_jcache_fixes_2	additional fixes necessary to complete full deployment of the Framework JCache offering
infra_13.2_synchronize_access_to_componentConsumers	make all methods that access componentConsumers ArrayList synchronized to eliminate ConcurrentModificationException
infra_13.2_bomba_cdx_framework_status_fix	track the status of the framework locally only rather than through the ConfigurationService. The older ConfigurationServiceSysManImpl seems to be very slow and unnecessary
infra_13.2_cert_xml_fix	Update host names in cert_genInput.txt
infra_13.4_UserSessionDescriptorImpl_fix_npe	Put protective code to guard against NullPointerException in UserSessionDescriptorImpl. Original problem on 11/10/2010 (prdbc84a/prdgc2a)
infra_13.4_blocking_reads	Update Grizzly / ORB to use blocking reads
infra_13.4_holiday_tteScraper	update Holiday property file through 2014 & add tteScraper script to scrape tradeeng log files for TTE class keys
infra_13.4_generAdmin_fixes	Extent map XML generator fixes

Appendix A - Database Changes

GC Database changes

There are no GC Table changes at this time.

BC Database changes

There are no BC Table changes at this time.

Appendix B – Installation order and pre-requisites

PAR User Role Changes

Day 1: Create and load new ACL. (ACL allows Market Maker roles to download products by reporting class, this is a call that Par needs to use).

Day 2: Change the PARWS user's role to Market Maker (This is the main acronym that all Workstations use).
- At this point in time none of the users will be impacted and they will still have the class display role associated with them.

Day 3: Update one of the user (for example: W101) using the SA GUI
- This will automatically change the users role in security service to Market Maker.

Day 4-n: Update 40 or so users each day until rollout is complete (Change some field and change it back so the update button gets highlighted).

Backout PAR User Role Changes

If for some reason we need to backout the role, we will need to change the PARWS's role to class display and update each of the modified users.

New User "XXP" to be added by membership

A new user XXP, similar in setup to XXH needs to be added by membership.

Verify using SA GUI that the user XXP has been added, is enabled and enablements are set.

Installation order

Note To install the BC we need to ensure that memory on all the bc's has been increased to handle the new memory requirement.
Also need to confirm we we can install both master and slave side bc on a weekend and backout the slave, this is to execute new-old and new – new failover with old and new style failover (so 3 failovers in total).

Day 0: Install Par clients with new Code base

Day 1: Install new ACL.

Day 1: Add new user XXP via Membership.

Day 1: Setup new SHM Alarms.

Day 1: Install Master GC1, 2 and 3 (Saturday).

Day 1: Make PAR User Role Changes (Change user PARWS and one Workstation login associated with that role.

Day 2: Install GC1,2 and 3 Slave side.

Day 2: Change all PAR users role to Market Maker..

Day 3: Install Master BC's (2-3 weeks, starting one Saturday).

Day 3: Install SA CAS's and SA GUI's.

Day 3: Install SHM's.

Day 4: Operator & Help desk training.

Day 4: Install Slave side BC's, *Can use new Failover procedure at this time.*

Day 5: Install Par Adapter & Par Svr's.

Day 5: Install CAS's (Internal & external) and FIX/Engines.

Day 5: Install GUI's.

Day 6: Turn on New BOB functionality.

Day 7: Install MDGC's

Day 7: Install LC's

Day 8: Minor on C2 BC's to change linkage correspondent from LKO to LKZ in the file **AutoLinkServiceHome.xml** for C2_MAIN session.

Appendix C – GC Installation Procedures

GC2 A/B Trade engine Installation Procedures

QA Steps

- At 3:15 have qa load the new software for release .

Infra group steps after End of all Sessions

- Load new ACL. (Need to load only once).

Server Group steps

- Most of the steps here can be done at 3:15.
- Only if needed fix the setContext file in /sbt/prod/tradeeng directory. After you login, if the setContext version has changed then the setContext version nbr needs to change .
- Change run_dir links for previous release.
- Change the run_dir link in /sbt/prod/tradeeng to point to the new release.
- Logout and log back in as tradengp.
- Run script $RUN_DIR/bin/genWatchedProcessList. Verify that $RUN_DIR/properties/ WatchedProcessListServer.out is generated and that all processes are listed correctly in this file.
- Do a diff against the old and new WatchedProcessListServer.out file to ensure they are the same. **DO NOT go any further is this does not work.**

GC1A/B Trade Engine Installation procedures

QA steps

- At 3:15 have qa load the new software using the QA setup steps.

CD Server group steps after end of all sessions

1. Change PARWS's role to Market Maker & then update one of the Par workstations user information (a dummy update) (The work station name will be provided at the 9:30 meeting).
2. Shutdown tradeengine on GC1.
3. Change run_dir links to point to the new CBOEDirect release.
4. Logout and log back in.
5. Update setContext in home directory if needed.
6. Start the GC processes.
7. **Master side only:**
   Load the Trading property "csv" file as explained (by running the command)
   ```
   TradingPropertyServiceClient setPropertyDefinitions
              $RUN_DIR/properties/TradingPropertyDefinitions.csv
   ```
8. **Master side only:**
   Add new routing properties, firm properties and trading properties as explained

   <u>Run the following command to add "Origin code Contingency mapping"</u>
   Run this command once for simple class SPX and once for complex class SPX (Get the class keys for SPX Options and Strategy by running the command `classKey –s SPX`)

   Copy the SPX option classkey into the following command:
   ```
   addOriginContingencyMappingRoutingProp -c <simpleClassKey>
   ```

   Copy the SPX strategy classkey into the following command:
   ```
   addOriginContingencyMappingRoutingProp -s <strategyClassKey>
   ```

   <u>Run the following command to add "Enable New BOB" property</u>
   This property should **<u>add & turn off</u>** New BOB for SPX

   ```
   addNewBOBEnabledProperties SPX OFF
   ```

   <u>Run the following command to add "Enable New BOB" property</u>
   Run this command once to add the default property fo rallied sal origin codes.

   ```
   addAllowedSalOriginCodesTradingProperties
   ```

9. Verify that all the new properties have been setup correctly – TODO (after SA GUI is installed) ?.

Verification after GC Upgrade

Note If you are installing the Slave side box then perform a fail-over so that the upgraded box becomes Master and then continue on with the remainder of the plan.

- ☐ Check all files (.log, .debug, .out, .err) for errors, exception's and high system alarms.
- ☐ Enable global external connections on the GC (**`globalExternalServices start`**).
- ☐ Start all sessions using the SA GUI
- ☐ Verify on prdgc01a/b that there are no products in NO_SESSION state.
- ☐ Use SAGUI to open test products on all the BC's. You can get the list of test classes from operations (This list is taped to one of the monitors in the basement)
- ☐ Login to 2 trader GUI's and bring up the market display window and status window) on each of the GUI's. NOTE: Status Window is a scrolling display that shows status messages for orders and quotes as and when they occur. (This window can be used to verify that OrderStatus and QuoteStatus events are working).
- ☐ Do test trades using quotes/orders on atleast one BC for all sessions.
- ☐ Verify that CurrentMarket (Mkt Bid and Mkt ask) is showing up on the market display window. This tests out that current market events are working.
- ☐ Verify that the last sale price and quantity fields are showing up on the Market display window. This tests out that Last Sale and Recap events are working.
- ☐ Verify that Order Status messages (New Order, Order filled are showing up on the Status window screen)
- ☐ Check all files on prdgc01 (.log, .debug, .out, .err) for errors, exception's and high system alarms.

Verifying MarketDataReportServer - for Last Sale and Recap processing

- ☐ Run script **createHalfHourlyReport – nw W_MAIN**.
- ☐ Verify that new report file is generated.
  log/HalfHourly_NewsWireReport_latest.xml.log
- ☐ Review the XML to see if the total volume is updated correctly.
- ☐ Repeat above for –hvol.
- ☐ Do not run for –opra as there is no way just to create report and not send. –opra option can be verified on Saturday testing only.
- ☐ Run ar MarketDataReportServer1 showTickerConsumerStats and verify number messages received.
- ☐ Do test trade on any test class for W_MAIN.
- ☐ Run above ar command again and verify total number of events received increased.
- ☐ Open trader gui and open DSP (Display Series Price) screen and do query for any W_MAIN product. Recap information should be displayed properly.
- ☐ Open trader gui and open VOL screen and do query for any W_MAIN class. VOL information.
- ☐ End all the sessions.
- ☐ Test Complete, Notify operations.

Backout procedures GC1A/B

To backout code shutdown tradeengine and flip the run_dir to the old release that was installed.

To backout user role changes, flip the role of user PARWS to CLASS DISPLAY and then do a dummy update on all the users that were converted..

Failover procedures GC1A/B

Follow regular GC failover procedures.

MDGC1 A/B Trade engine Installation Procedures

QA Steps

- ☐ At 3:15 have qa load the new software for release .

Server Group steps

- ☐ Shutdown tradeengine processes
- ☐ Most of the steps here can be done at 3:15.
- ☐ Only if needed fix the setContext file in /sbt/prod/tradeeng directory. After you login, if the setContext version has changed then the setContext version nbr needs to change .
- ☐ Change run_dir links for previous release.
- ☐ Change the run_dir link in /sbt/prod/tradeeng to point to the new release.
- ☐ Logout and log back in as tradengp.
- ☐ Start Tradeengine processes.

Verification (ONE_MAIN and W_MAIN) – XTP Underlying feed

- ☐ Verify all .log, .err, .out, .debug for exceptions and alarms.
- ☐ Put all W_MAIN products in pre-open state (Be care full when doing this on a weekday).
- ☐ Start tips replay.
- ☐ Verify products are moving to opening rotation.
- ☐ Enter a quote on one real product , verify MDH has latest underlying data.

Verification (ONE_MAIN and CFE_MAIN) – Bookdepth Feed

- ☐ Verify all .log, .err, .out, .debug for exceptions and alarms.
- ☐ Put all Test classes for ONE_MAIN and CFE_MAIN products in pre-open
- ☐ Enter Quotes.
- ☐ Using ar command on CfnAdapter1 enable refresh of Bookdepth on all sessions.
- ☐ Using cfnAdmin monitor all the 12 Multicast lines (6 fro CFE primary and backup and 6 for ONE_MAIN primary and backup). Full refresh of bookdepth is done every few seconds, you will see your test classes having data in them.

Backout procedures MDGC1A/B

To backout code, shutdown tradeengine and flip the run_dir to the old release that was installed.

Failover procedures MDGC1A/B

Follow regular MDGC 1 failover procedures.

MDGC 2 A/B Trade engine Installation Procedures

QA Steps

- ☐ At 3:15 have qa load the new software for release .

Server Group steps

- ☐ Shutdown tradeengine processes
- ☐ Most of the steps here can be done at 3:15.
- ☐ Only if needed fix the setContext file in /sbt/prod/tradeeng directory. After you login, if the setContext version has changed then the setContext version nbr needs to change .
- ☐ Change run_dir links for previous release.
- ☐ Change the run_dir link in /sbt/prod/tradeeng to point to the new release.
- ☐ Logout and log back in as tradengp.
- ☐ Start Tradeengine processes.

Verification

- ☐ Verify all .log, .err, .out, .debug for exceptions and alarms.
- ☐ Open all test classes in W_MAIN session.
- ☐ Have operations bring up an RCN screen.
- ☐ Enter a quote on every single Test class.
- ☐ Verify on an RCN screen if quotes have been received on all test classes.
- ☐ If an RCN is not available you can use mdbAdmin to monitor the data that is broadcast on the broadcast udp ports. NOTE: data is broadcast on 2 networks, you should monitor both the networks. (Monitor some classes on one network and other classes classes on the 2nd network).

Backout procedures MDGC2A/B

To backout code, shutdown tradeengine and flip the run_dir to the old release that was installed.

Failover procedures MDGC2A/B

Follow regular MDGC 2 failover procedures.

GC - Saturday verification after upgrade

#	Description	PASS/FAIL
1	loadOpenInterest test (performed by Ops at startup- requested at weekend test meeting)	
2	Product download test – CAS (performed by Ops at startup)	
3	Restart all CAS & Fix Engines. Verify CAS startup time. (performed by Ops at startup)	
4	Start all sessions using the SA-GUI (performed by Ops at startup)	
5	Verify all products are assigned to sessions and no product is in NO_STATE (Ops at startup)	
6	Run ITG Checkout scrpt on all BC's (performed by Ops at startup)	
7	Transition all products for all sessions to PRE-OPEN state.	
8	Run the XTP replay, replaying the Friday 8:29-8:45 traffic at a 1.5 times rate. Verify that all W_MAIN classes transitioned to OPENING-ROTATION, verify ticker and recap on GUI, Verify underlying price in MDH. Verify broadcast to a PDS. Verifies Data from MDGC1.	
9	OPEN all products for all sessions.	
10	Kill a CAS to verify users are logged out - SMS test	
11	Enter Quote - Book Depth update (dynamic) - CFE/ONE_MAIN Verify that data goes ot of CfnAdapter1 on mdgc01 (3 outbound lines for CFE and 3 outbound lines for ONE). **ar cfnAdmin monitor raw CFE_MAIN C 1**	
12	Generate and verify Half Hourly reports for News Wire and HVOL. Actually sending the file to OPRA can only be verified on the weekend. Only on the weekend can you use the –opra option. * Verify hourly reports are generated every half an hour by Control-M * Run script **createHalfHourlyReport – nw W_MAIN**. * Verify that new report file is generated. log/HalfHourly_NewsWireReport_latest.xml.log * Repeat above for –hvol.	
13	Examine global log to verify open interest was loaded	
14	Using SA_GUI, spot check some open interest for some of the products. There should be non-zero values for most of the products. It is OK if a few of them have zero values because it might be a result of new series additions.	
15	Run **ar MarketDataReportServer1 showStatistics** and verify number messages received	
16	Do test trade on any production class for W_MAIN. Note : SATURDAY TESTING ONLY	
17	Repeat step 16 and verify total number of events received increased	
18	Repeat step 13 and verify the reported numbers have increased	
19	Pause XTP Replay.	
20	Perform four BCXX failovers with GC01a is master. Where XX is a single hybrid, CFE, ONE, and STOCK BC.	

21	Run ITG Checkout scripts on the failed over BCs (only). While running proceed to step 23.	
22	Verify that data CM , LS and Product states are going out to PDSs. The PDSs are verified as follows: Verify the PDS Overhead is updating with values from the class being used. Quote and last sale information should change. This verifies the MDB broadcast. Depending on the type of display OPS has configured on their test PDS Overhead, the last sale might not show up. Have them switch screen types until they find the one that shows the last sale. **Make sure XTP replay is NOT running during the ITG checkouts**. XTP Replay can cause spooling and there would be a possibility of dropped quotes/trades which could cause the test to appear to fail.	
23	Failover MDGC01 and MDGC02 with new GC01a as master.	
24	Verify CFE/ONE_MAIN book depth data is being published by repeating step 12.	
25	FE failover test with GC01a as master. KILL FE03	
26	Restart CAS2011. Verify the cas reinitializes. Run ITG Checkouts on any of the following BC/TE combinations that use FE03. BC04/TS-4 (cas2011) BC10/TS-3 (cas2011) BC30/TS-2 (cas2011) BC82/TS-1 (cas2011) BC90/TS-3 (cas2011) BC93/TS-4 (cas2011) BC98/TS-1 (cas2011)	
27	Start XTP Replay (1.5 rate(. Open the *Market Display For Underlying* window and verify ticker and recap.	
28	VERIFY BROADCAST TO A PDS. THE PDSS ARE VERIFIED AS FOLLOWS:Verify the PDS Overhead is updating with values from the class being used. Quote and last sale information should change. This verifies the MDB broadcast. Depending on the type of display OPS has configured on their test PDS Overhead, the last sale might not show up. Have them switch screen types until they find the one that shows the last sale.	
29	GC01 Fail over (new to old)	
30	Pause XTP Replay.	
31	RUN CMi and FIX ITG Checkout scripts (all BCs) after the fail over.	
32	Verify Failover times to see how long it takes to do the complete failover (Stop + goMaster + Pre-open products)	
33	Close->PreOpen->Open all products in all sessions. Time the transition from Close to PreOpen	
34	Bounce a CAS to verify it “re-inits”	
35	Verify the CAS startup time against the previous morning startup time using the CAS Start Times script.	
36	Failover GC2.	
37	Run any special procedures provided by infra group on how to verify gc2 failover.	

Appendix D – BC Installation procedure

QA steps

- ☐ At 3:15 have qa load the new software.

Server group steps after end of all sessions

1. Verify that the BC table changes have been done as explained above (None identified so far).
2. Verify that the default routing properties and trading properties have been set correctly as explained above.
3. Shutdown tradeengine on the BC being installed (tradengp and tradengh login).
4. Install new software CBOEDIR.xxx release, change run_dir links .
5. Start the BC processes (tradengh first and then tradengp logins).
6. Enable business external connections on the master bc.
7. Start all sessions, do a quick quote and order test on one test class on the affected BC.

Verification

- ☐ Check all files (.log, .debug, .out, .err) for errors, exceptions and high system alarms.
- ☐ Make sure all initialization is complete on all processes.
- ☐ Open sessions associated with this bc pair and send a few test orders & quotes, fill one test order and leave a couple of orders in the book. All Test orders should be for customer origin.
- ☐ When installing master side box, please verify if the distributed caches are updated properly

Note At this point in time the Slave side caches have not yet been installed, so we can only check the distributed cache on the master side).

On Master TS run the following commands to verify the counts of different caches

a) `ar HybridTradeServer1 distributedCache TradingProductCache`

b) `ar HybridTradeServer1 distributedCache GenericCache`

c) `ar HybridTradeServer1 distributedCache OrderCache`

On Master OHS run the following commands to verify the counts of different caches

xvii. `ar HybridTradeServer1 distributedCache GenericCache`

d) `ar HybridTradeServer1 distributedCache OrderCache`

- [ ] When installing the slave side box please verify that the order caches are in synch by running the following commands

   a) **`status –count (run this one either on master or slave)`**

Note The command above will display the order counts on all Trade Servers and the OH Servers on that bc for both master and slave side boxes. If the count of orders is the same on master and slave side we are ok, otherwise there is something wrong with the installation.

- [ ] **On Master and slave TS's run the following commands to verify the counts of different caches are in synch with each other**

   `a) ar HybridTradeServer1 distributedCache TradingProductCache`

   b) **`ar HybridTradeServer1 distributedCache GenericCache`**

On Master & Slave OHS run the following commands to verify the counts of different Caches are in synch with each other.

   `a) ar OHServerHybrid distributedCache GenericCache`

- [ ] If you are installing the Slave side box then perform a fail-over so that the upgraded box becomes Master and then continue on with the remainder of the plan.

Note We need to do the failover twice, Old Failover style and new Fast Failover before each failover we need to ensure that the caches are in synch while there are orders in the book.

- [ ] Run the command **`businessExternalServices check`** to make sure that remote connections are established for the adapters on that bc.
- [ ] Verify on prdgc01a/b that there are no products in NO_SESSION state.
- [ ] Use SAGUI to open test products associated with the affected BC's . You can get the list of test classes from operations (This list is taped to one of the monitors in the basement)
- [ ] Login to 2 trader GUI's and bring up the market display window and status window) on each of the GUI's. NOTE: Status Window is a scrolling display that shows status messages for orders and quotes as and when they occur. (This window can be used to verify that OrderStatus and QuoteStatus events are working).
- [ ] Run the ITG checkout script on all affected bc's only.
- [ ] Verify that last sale from affected bc's are showing up on MDRS that runs on the global server.
- [ ] Check all files on the affected BC's (.log, .debug, .out, .err) for errors, exceptions and high system alarms.
- [ ] Run the ar command **`hsAdmin –c stats –p HybridHistoryServer1`** to verify if counts on the HybridHistoryServer are increasing.
- [ ] Run the ar command **`hsAdmin –c stats –p HybridTradeServer1`** to verify that counts on the TradeServer are increasing (which means that the TradeServer is sending data to the history server).

Note Some times after end of session the external applications will not allow connections afer hours, so the only way to verify is by looking at the log file to make sure that the program's are making an attempt to connect to the remote system on correct ip address and correct port nbrs.

Final verification

- ☐ Close all products in all sessions using the SA GUI (Pick the tab to close ALL the products).
- ☐ Verify memory usage for **all processes** on the BC and Garbage Collection activity by comparing with the OLD and NEW .out files.
- ☐ End all the sessions
- ☐ If you started this upgrade on the Slave side perform a fail-over so that the upgraded box becomes Slave now.
- ☐ Test Complete, Notify operations.

Failover

- ☐ See operator procedures on how to failover BC's or list instructions specific to your release here.

Fallback

- ☐ Revert the run_dir link back to the old release and restart tradeengine processes to verify that the release has been backed out.

Saturday BC verification after upgrade

#	Description	PASS/FAIL
	You will need 2 trader gui's and 1 sa gui - login thru all these guis before you start the test. XtpReplay data should be captured on a production day between 8:29-8:45. This applies only if installing on a W_MAIN or W_STOCK BC.	
1	Pre-requisites – Ops has brought up system and successfully ran checkouts	
2	Verify on prdgc01a/b that there are no products in NO_SESSION state.	
3	Ask Ops to verify the "checkout results" e-mail that is sent to the CCS e-mail group when checkouts complete. There should be 3 e-mails for CMI and 3 e-mails for FIX.	
4	Verify XTP Data is being received by the trade servers, Underlying recap (W_MAIN), BOTR exchange indicators (W_MAIN and W_STOCK using a production class) - Verify using ar commands on the bc (on the Trade Servers)..	
5	Using a Trader GUI, do MDH Queries and verify NBBO and exchange indicators.	
6	Run the following commands to get counts. These commands will be run again after ITG checkouts. The counts should increase. Run **ar MarketDataReportServer1 showStatistics** on the GC and note the number of messages received. Run "hsAdmin –c stats –p HybridHistoryServer1" to get count Run **"**`hsAdmin –c stats –p HybridTradeServer1`**"** to get count	
7	Run ITG checkouts for the BC being installed.	
8	WHILE ITG CHECKOUTS ARE RUNNING - Verify current market, last sale, trade reports, fill reports over all external connections. Enter a quote, order, and trade for a production class for the W_MAIN, W_STOCK, W_ONE, W_CFE sessions. **CTMr** – verify there are no un-acked trade reports. This can be verified via the following command: (**TO DO: Insert command here)** **COPP** – This is for W_MAIN. verify the COPP GUI shows a queue being build for the OPRA lines. This will be true if COPP was brought up *without* OPRA being live and *without* the OPRA simulator being up. These conditions result in no place for COPP to send the data and it will queue inside COPP. **FOPP-ONE** – This is for ONE_MAIN. verify the FOPP-ONE GUI shows an increase in the number of messages that were sent out the outbound lines. **FOPP-CFE** – This is for CFE_MAIN. verify the FOPP-CFE GUI shows an increase in the number of messages that were sent out the outbound lines. **STOPP** – This is for W_STOCK. verify the STOPP GUI shows a queue being build for the outbound lines. This will be true if STOPP was brought up *without* STIC/NASDAQ being live and *without* the STOPP simulator being up. These conditions result in no place for STOPP to send the data and it will queue inside STOPP. **PDS** - This instruction only applies if you are installing a W_MAIN BC. Verify the PDS Overhead is updating with values from the class being used. Quote and last sale information should change. This verifies the MDB broadcast. Depending on the type of display OPS has configured on their test PDS Overhead, the last sale might not show up. Have them switch screen types until they find the	

	one that shows the last sale. Make sure XTP replay is NOT running during the ITG checkouts. XTP Replay can cause spooling and there would be a possibility of dropped quotes/trades which could cause the test to appear to fail.	
9	Run the following commands to get counts. The counts should have increased from the values obtained before the ITG Checkout Run MDRS Admin Request on GC to show “last sale” count Run “hsAdmin –c stats –p HybridHistoryServer1” to get count Run “`hsAdmin –c stats –p HybridTradeServer1`” to get count	
10	If this is a BOB BC installation Operations or on site support staff need to enter a manual price report in for one of the products on the Trader GUI. The last sale column should be updated. Check, MDH, there should be an entry for this last sale.	
11	**Verify Order Cache is in synch before failover (This requires master and slave to be upgraded with the same 8.6 release – If not done on Saturday then this step needs to be done on the day when the slave bc is installed)** Run the command ‘`status –count’ once` to display count of orders in the distributed cache on both master and slave side. Verify that the counts of orders are same on both master and slave bc’s.	

12	**Verify all other Caches are in synch before failover (This requires master and slave to be upgraded with the same 8.6 release - If not done on Saturday then this step needs to be done on the day when the slave bc is installed)**
	On both master and slave bc’s run the following commands to compare the counts of individual caches
	On TS’s run the following commands
	`a) ar HybridTradeServer1 distributedCache TradingProductCache`
	`b) ar HybridTradeServer1 distributedCache GenericCache`
	On OHS’s run the following commands
	`b) ar OHServerHybrid distributedCache GenericCache`	

13	**BC Failover test, We need to repeat the section below for** a) New BC – Old BC : Old Style Failover b) New BC – New BC: Old Style Failover (can be done when slave is installed) c) New BC – New BC: New Style Failover (can be done when Slave is installed)	
14	For the BC being installed, use the procedures as listed in the operator procedures and have ops execute the BC failover over procedures.	
15	Time the fail over, it should take roughly 3 minutes for Old Failover & 200 Millis for Fast Failover.	
16	After the fail over run the ITG checkout script on all BC’s.	
17 18 19	Enter Manual Price Report – Verify Last Sale Update – Verify MDH entry **<u>If installing on a BOB BC: (otherwise, skip this step)</u>** Enter Manual Quote – Verify Current Market Update Enter Manual Price Report – Verify Last Sale Update – Verify MDH entry Open DSP (Display Price Series) screen and check information displayed for product. The DSP screen is opened on the Trade GUI using a login that has a *price reporting* roll. Close the products, End the sessions and verfy that all the sessions have ended successfully.	

Appendix E - LC Installation procedures

QA steps

- ☐ At 3:15 have qa load the new software.

Server group steps

- ☐ Notify help desk that LC will be upgraded and that next day they may need to remount the log directories on their CMC Viewer PC's.
- ☐ Master or Slave side can be upgraded after 3:15. There is no need to wait for End Of Session.
- ☐ If installing the master side box shutdown tradeengine processes on the Slave Side first then shutdown tradeengine processes on the master side.
- ☐ If installing the slave side then Shutdown tradeengine processes on the Slave side.
- ☐ Change run_dir links to point to the new release.
- ☐ Only if needed fix the setContext file in /sbt/prod/tradeeng directory. After you login, if the setContext version has changed then the setContext is not run and you will not be able to start any process.

Note Delete orun_dir and move run_dir to orun_dir. Helpdesk needs "orun_dir" to look at old Linkage log files.

- ☐ Logout and log back in as tradengp.
- ☐ Have operations bring up tradeengine processes using PATROL
- ☐ If this is the slave box then failover and run thru the verification steps listed below.

Verification

- ☐ Check all files (.log, .debug, .out, .err) for errors, exceptions and high system alarms.
- ☐ Make sure all initialization is complete on all processes.
- ☐ If you are installing the slave side box then installation is complete. (Just make sure operations runs the slave box in master mode the next day).
- ☐ Verify memory usage for **all processes** on the LC's and Garbage Collection activity by comparing with the OLD and NEW .out files.
- ☐ Just pick a test class and send the out bound Order, it will get cancelled in the "Fix Linkage Adapter" with a reason code of Away Exchange cancel (199).

Appendix F – Details on how to execute various commands

Note These commands are to be used during development installs only and not in production.

ACL update to allow Par to add Orders

There is an ACL change tied to this release. When the new code base is installed, server INFRA should load the new ACL.

The ACL list has been updated to allow par users to add orders via PAR workstation. The method "**getProductsForReportingClassSymbol**" needs to be updated to allow market makers access. The vob is controlled by CAS and lives in the infra directory under name ACL.csv

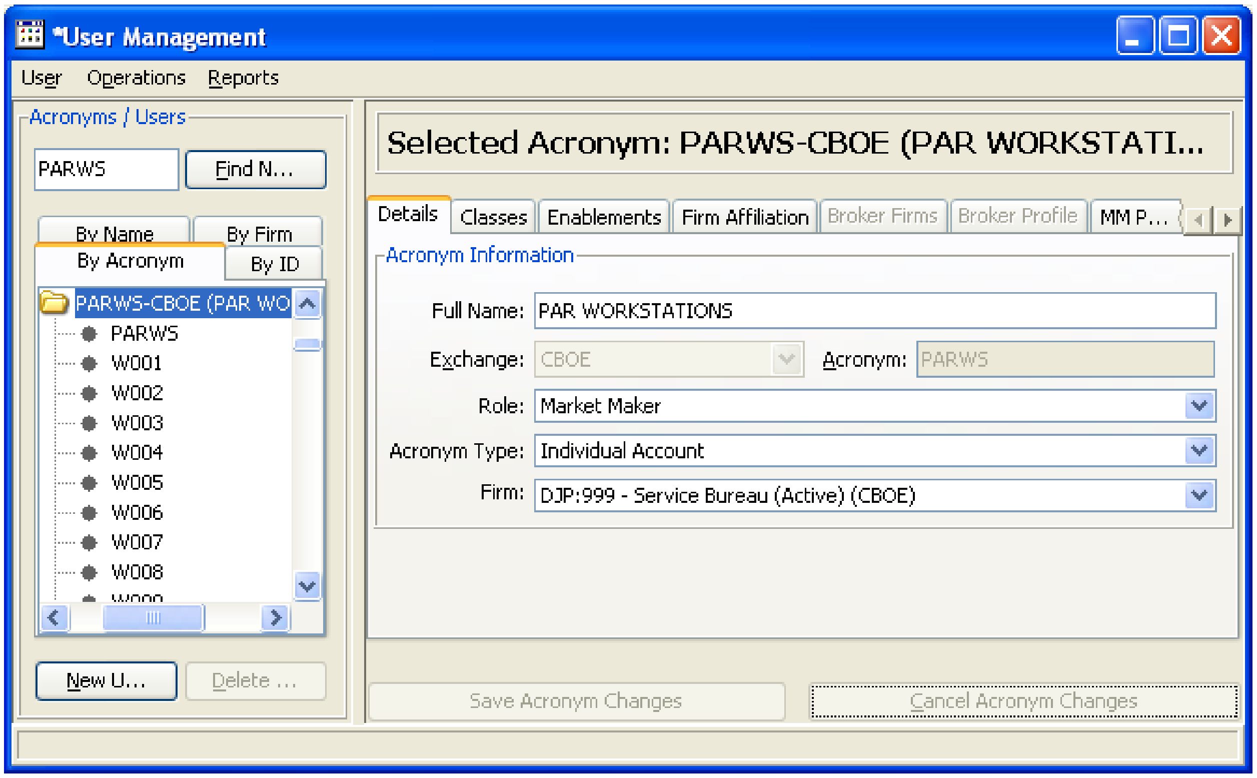
Loading new ACL

1. FTP new ACL.csv to tmp directory of infra Global
2. updateRoles -R INFRA_THIRTEEN_FOUR_13.1.16.5.1 /sbt/test23/infra/INFRA_THIRTEEN_FOUR_13.1.16.5.1/tmp/ACL.csv

   (above example is for infra in test23. first parm –R, Second (infra label), Third (path to ACL)

Changing PAR User roles

1. Change (or ADD) PAR User with Market Maker Role Type.
2. Select PARWS Acronym in user service
3. Change ROLE: From Class Display To Market Maker.



PARWS

PARWS-CBOE (PAR WO
- PARWS
- W001
- W002
- W003
- W004
- W005
- W006
- W007
- W008

Selected Acronym: PARWS-CBOE (PAR WORKSTATI...

Full Name: PAR WORKSTATIONS

Exchange: CBOE Acronym: PARWS

Role: Market Maker

Acronym Type: Individual Account

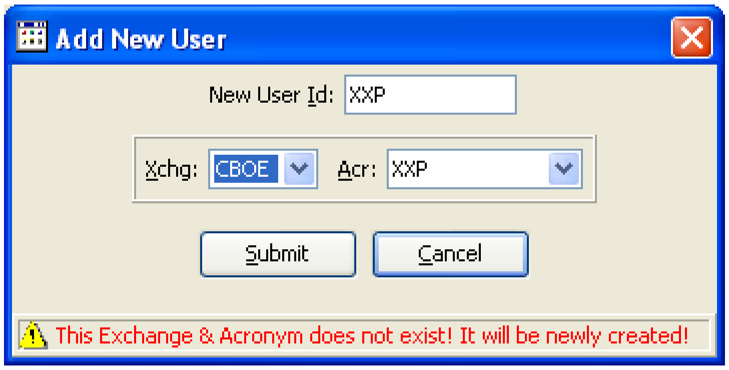
Firm: DJP:999 - Service Bureau (Active) (CBOE)

Note Individual logins under PARWS will not be updated until next day in development. To activate an individual Login immediately you have to reset the individual login by resaving it.

Adding user XXP to User Service

XXP needs to be added to the user Service. If not an exception will be thrown in OHS. The MM are validated when a tranction comes from a PAR workstion. XXP is now a valid Broker. Use the same settings for XXP as XXH. XXH is already defined in User Service.

A.) Select User → new user

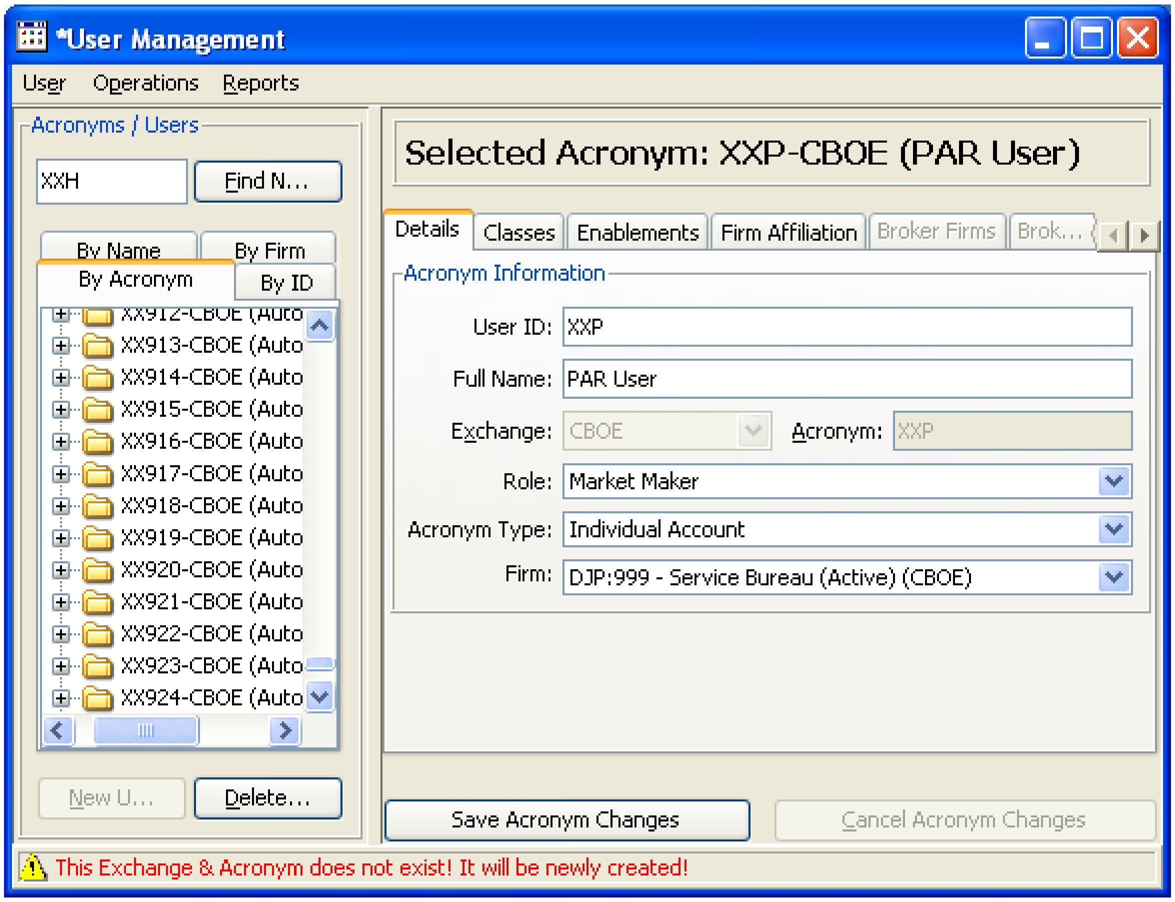


New User Id: XXP

Xchg: CBOE Acr: XXP

This Exchange & Acronym does not exist! It will be newly created!

B) Select Details.



XXH

XX912-CBOE (Auto
XX913-CBOE (Auto
XX914-CBOE (Auto
XX915-CBOE (Auto
XX916-CBOE (Auto
XX917-CBOE (Auto
XX918-CBOE (Auto
XX919-CBOE (Auto
XX920-CBOE (Auto
XX921-CBOE (Auto
XX922-CBOE (Auto
XX923-CBOE (Auto
XX924-CBOE (Auto

Selected Acronym: XXP-CBOE (PAR User)

User ID: XXP

Full Name: PAR User

Exchange: CBOE Acronym: XXP

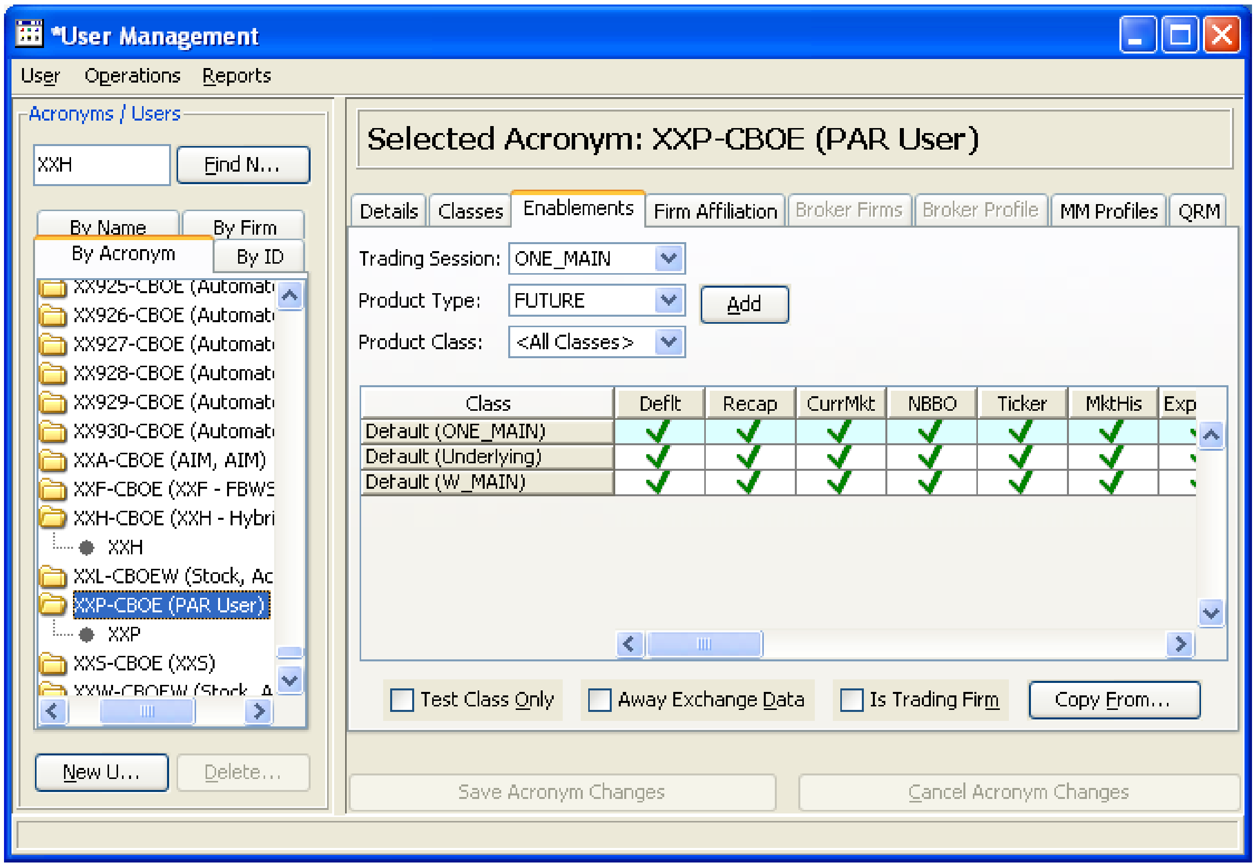
Role: Market Maker

Acronym Type: Individual Account

Firm: DJP:999 - Service Bureau (Active) (CBOE)

This Exchange & Acronym does not exist! It will be newly created!

C.) Select Enablements. Add ONE_MAIN, Underlying, and W_MAIN. Leave defaults.



XXH

XX925-CBOE (Automat
XX926-CBOE (Automat
XX927-CBOE (Automat
XX928-CBOE (Automat
XX929-CBOE (Automat
XX930-CBOE (Automat
XXA-CBOE (AIM, AIM)
XXF-CBOE (XXF - FBWS
XXH-CBOE (XXH - Hybri
XXH
XXL-CBOEW (Stock, Ac
XXP-CBOE (PAR User)
XXP
XXS-CBOE (XXS)
XXW-CBOEW (Stock, A

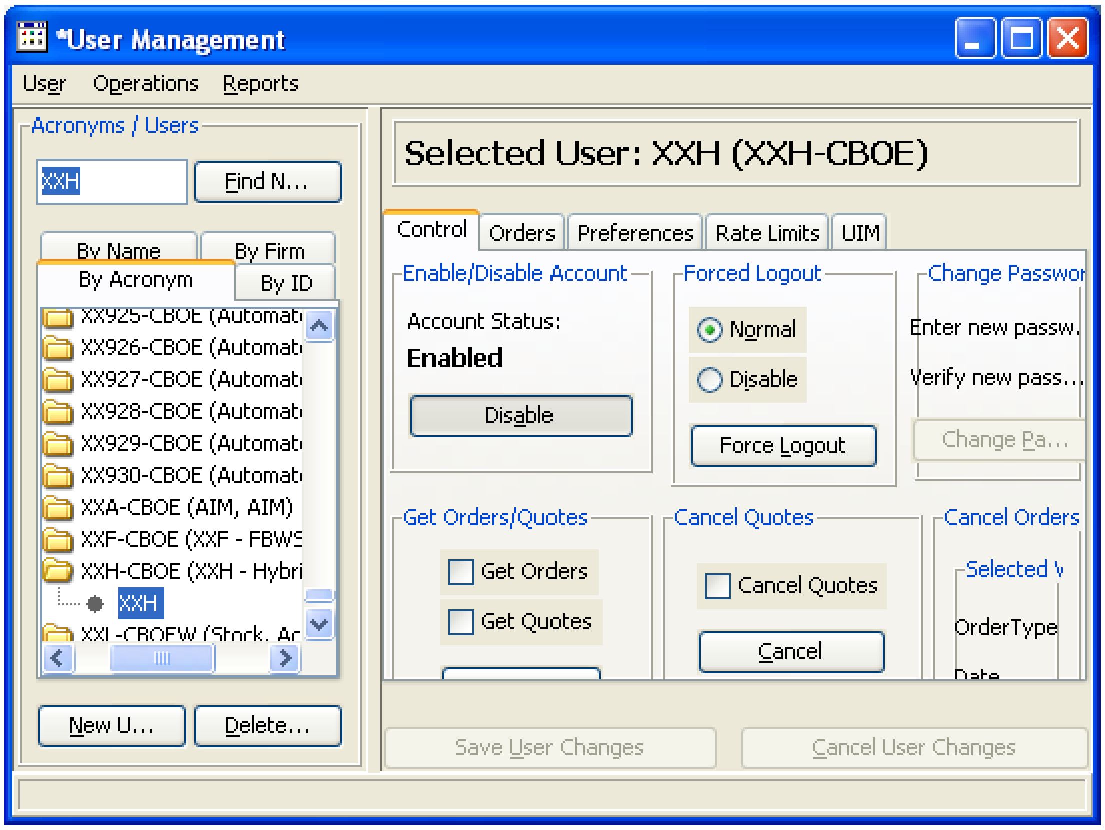
Selected Acronym: XXP-CBOE (PAR User)

Trading Session: ONE_MAIN
Product Type: FUTURE
Product Class: <All Classes>

Class	Deflt	Recap	CurrMkt	NBBO	Ticker	MktHis	Exp
Default (ONE_MAIN)	✓	✓	✓	✓	✓	✓	✓
Default (Underlying)	✓	✓	✓	✓	✓	✓	✓
Default (W_MAIN)	✓	✓	✓	✓	✓	✓	✓

☐ Test Class Only ☐ Away Exchange Data ☐ Is Trading Firm

D.) Select XXP Login. Under Control tab select Enable.



Selected User: XXH (XXH-CBOE)

Account Status: **Enabled**

[✓ Normal] Disable

☐ Get Orders

☐ Get Quotes

☐ Cancel Quotes

Change Allocation Strategies for SPX

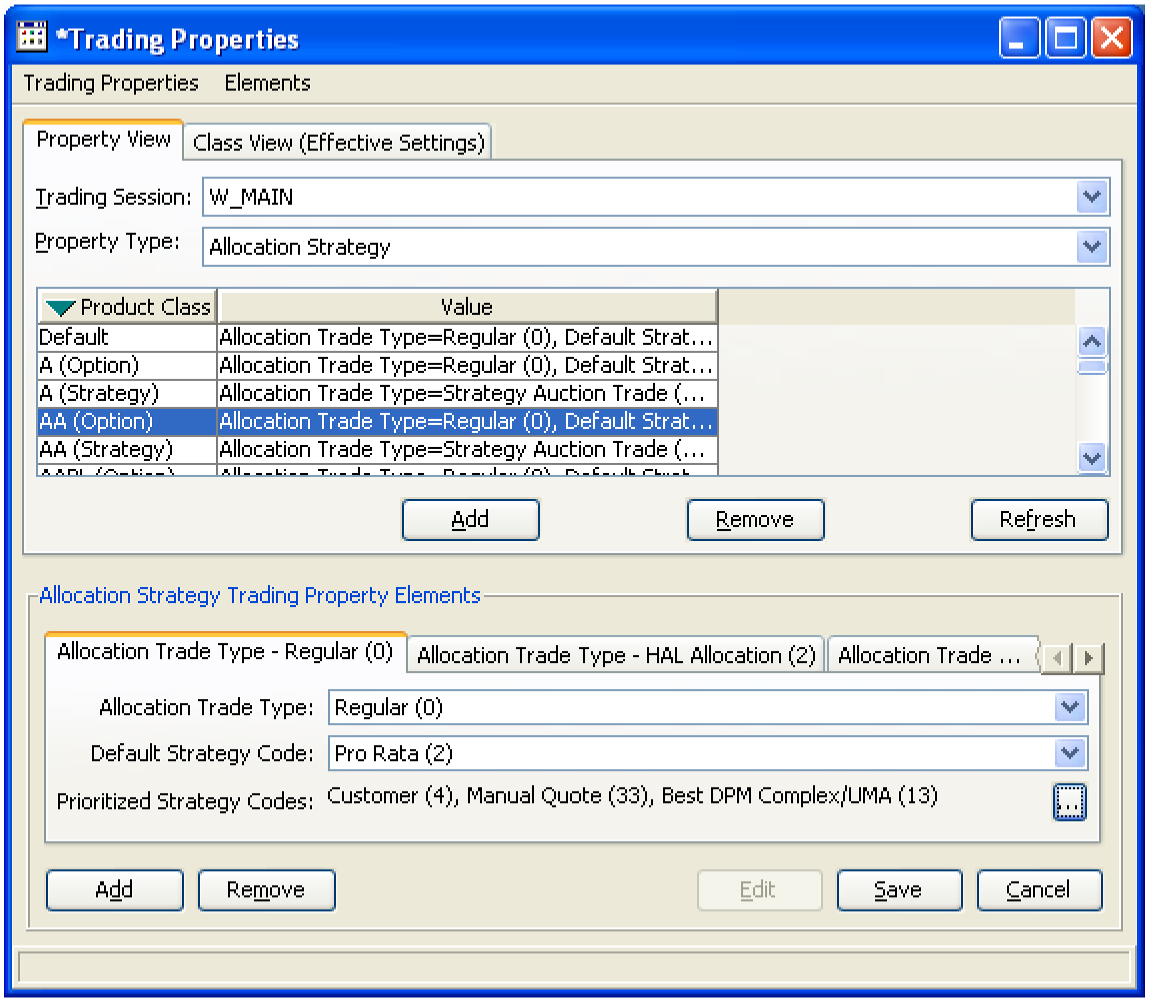
Regular Allocation

Regular(0)

Pro Rata(2)

Customer(4), Manual Quote(33), Best DPM Complex/UMA(13)

In Below example using AA for SPX.



*Trading Properties

Trading Session: W_MAIN

Property Type: Allocation Strategy

Product Class	Value
Default	Allocation Trade Type=Regular (0), Default Strat…
A (Option)	Allocation Trade Type=Regular (0), Default Strat…
A (Strategy)	Allocation Trade Type=Strategy Auction Trade (…
AA (Option)	Allocation Trade Type=Regular (0), Default Strat…
AA (Strategy)	Allocation Trade Type=Strategy Auction Trade (…

Allocation Strategy Trading Property Elements

Allocation Trade Type - Regular (0) | Allocation Trade Type - HAL Allocation (2) | Allocation Trade …

Allocation Trade Type: Regular (0)

Default Strategy Code: Pro Rata (2)

Prioritized Strategy Codes: Customer (4), Manual Quote (33), Best DPM Complex/UMA (13)

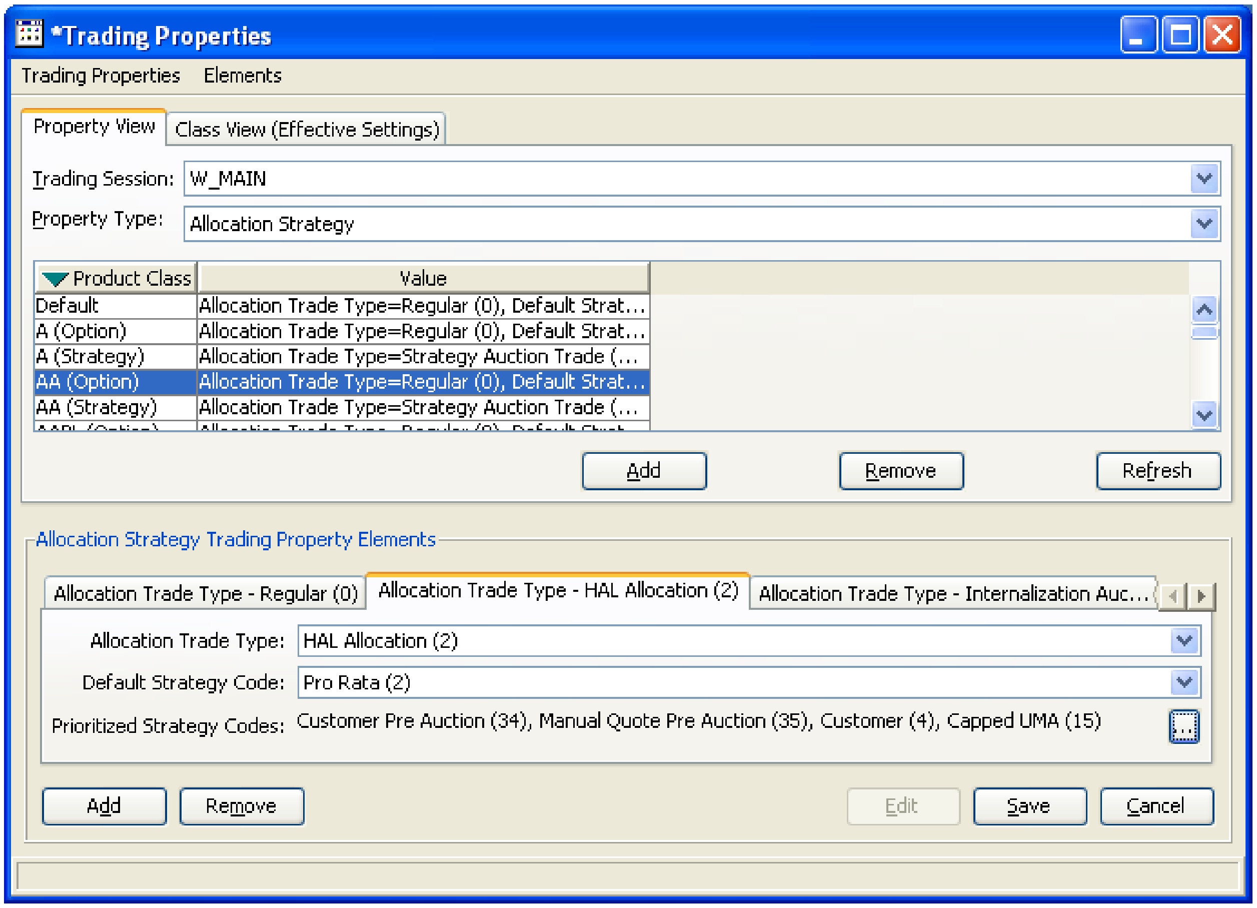
Hal Allocation

Hal Allocation(2)

Pro Rata(2)

Customer_pre_auction(34), Manual_Quote_Pre_Auction(35), Customer(4), Capped UMA(15)

In Below example using AA for SPX


*Trading Properties
Trading Properties
Elements
Property View
Class View (Effective Settings)
Add
Remove
Refresh
Allocation Strategy Trading Property Elements
Allocation Trade Type - Regular (0)
Allocation Trade Type - HAL Allocation (2)
Allocation Trade Type - Internalization Auc...
Add
Remove
Edit
Save
Cancel

Trading Session: W_MAIN

Property Type: Allocation Strategy

Product Class	Value
Default	Allocation Trade Type=Regular (0), Default Strat...
A (Option)	Allocation Trade Type=Regular (0), Default Strat...
A (Strategy)	Allocation Trade Type=Strategy Auction Trade (...
AA (Option)	Allocation Trade Type=Regular (0), Default Strat...
AA (Strategy)	Allocation Trade Type=Strategy Auction Trade (...

Allocation Trade Type: HAL Allocation (2)

Default Strategy Code: Pro Rata (2)

Prioritized Strategy Codes: Customer Pre Auction (34), Manual Quote Pre Auction (35), Customer (4), Capped UMA (15)

Script to define property “NewBOBOriginCodeContingencyMapping”

A script has been written to automatically update routing property ““New BOB” Origin Code Contingnecy Type Mapping property”.

Usage:

addOriginContingencyMappingRoutingProp [-c] simpleClassKey [-s] strategyClassKey [-h]
-c : The simple classKey of the Bob Class
-s : The strategy classKey of the Bob Class
-h : Help

Origins it will add:

This script will add origins B,F,I,X,M,N,K with the contingencies that are allowed to route to trade engines.

B,F,IX -→ will allow IOC and OPG
M → will allow IOC,OPG, and AR
N → OPG

K- NONE (or no contingency) this should also only be allowed from PAR terminal for Manual Quote

These will be added for both Simple and Complex.

Script to define property “New BOB Enabled”

The below script will set class SPX (BOB) to DISABLED for “New BOB” processing.

addNewBOBEnabledProperties SPX OFF.
Usage:

addNewBOBEnabledProperties <comma separated class symbol> [turn ON|OFF]
Argument 1 <comma separated class symbol>: any valid comma separated option class symbols
Argument 2 [turn ON|OFF]: this an optional parameter by default is OFF, valid values are => ON, OFF

E**xample: addNewBOBEnabledProperties SPX OFF**

Script to set Allowed Origins For SAL

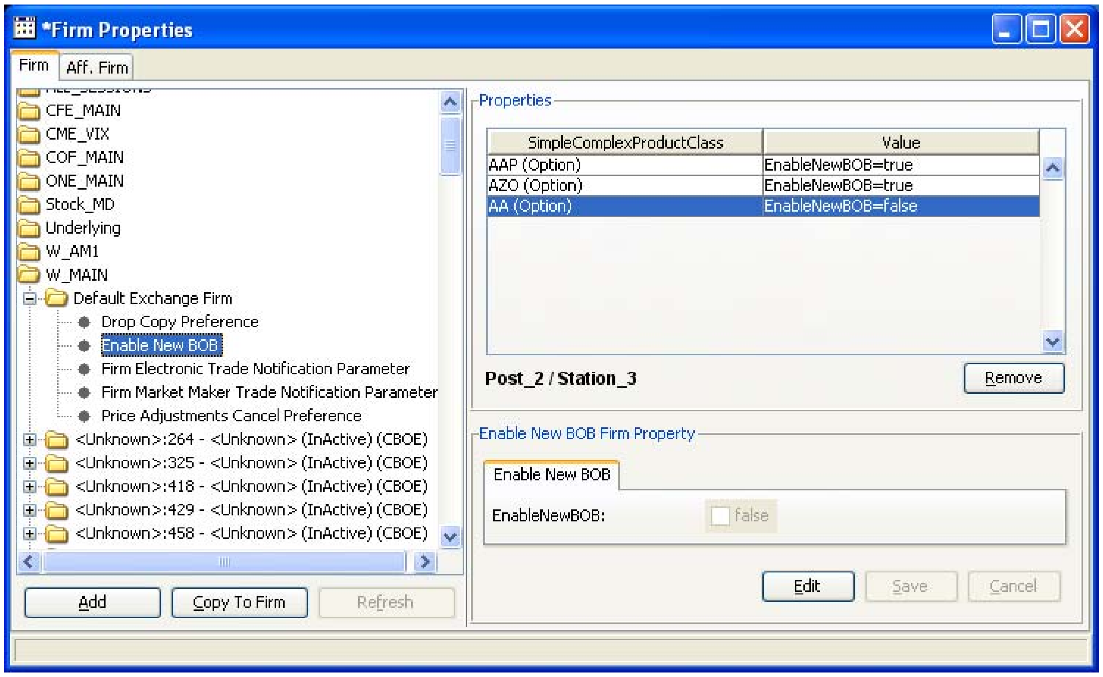
The below script will set the orgin codes that will be SAL if appropriate.
addAllowedSalOriginCodesTradingProperties

The script will add origin codes C, F, and B at default and set them to False.
W_MAIN will be set the C = True, W = True. (only "New BOB" will check this property)
C2_MAIN C, F, B will be set to true.

Verifying the new properties using SA GUI

Verifying "Enable New BOB"

Got to firm properties → W_MAIN → select Default Exchange Firm → Enable "New BOB"

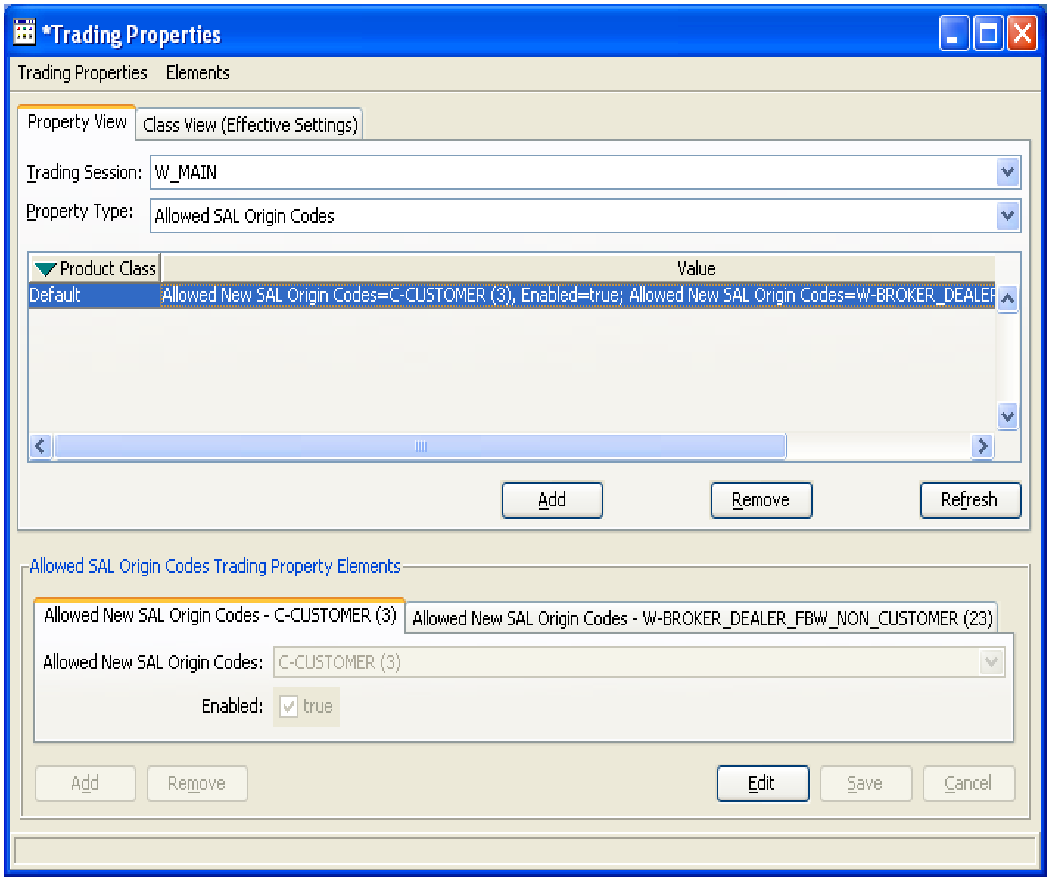

*Firm Properties
Firm
Aff. Firm
CFE_MAIN
CME_VIX
COF_MAIN
ONE_MAIN
Stock_MD
Underlying
W_AM1
W_MAIN
Default Exchange Firm
Drop Copy Preference
Enable New BOB
Firm Electronic Trade Notification Parameter
Firm Market Maker Trade Notification Parameter
Price Adjustments Cancel Preference
<Unknown>:264 - <Unknown> (InActive) (CBOE)
<Unknown>:325 - <Unknown> (InActive) (CBOE)
<Unknown>:418 - <Unknown> (InActive) (CBOE)
<Unknown>:429 - <Unknown> (InActive) (CBOE)
<Unknown>:458 - <Unknown> (InActive) (CBOE)
Add
Copy To Firm
Refresh
Properties
Post_2 / Station_3
Remove
Enable New BOB Firm Property
Enable New BOB
Edit
Save
Cancel

SimpleComplexProductClass	Value
AAP (Option)	EnableNewBOB=true
AZO (Option)	EnableNewBOB=true
AA (Option)	EnableNewBOB=false

EnableNewBOB: ☐ false

Verifying Allowed SAL Origin Codes

Check SA gui trading properties after run. Allowed SAL Origin Codes should look as follows:


*Trading Properties
Trading Properties
Elements
Property View
Class View (Effective Settings)
Add
Remove
Refresh
Allowed SAL Origin Codes Trading Property Elements
Allowed New SAL Origin Codes - C-CUSTOMER (3)
Allowed New SAL Origin Codes - W-BROKER_DEALER_FBW_NON_CUSTOMER (23)
Add
Remove
Edit
Save
Cancel

Trading Session: W_MAIN

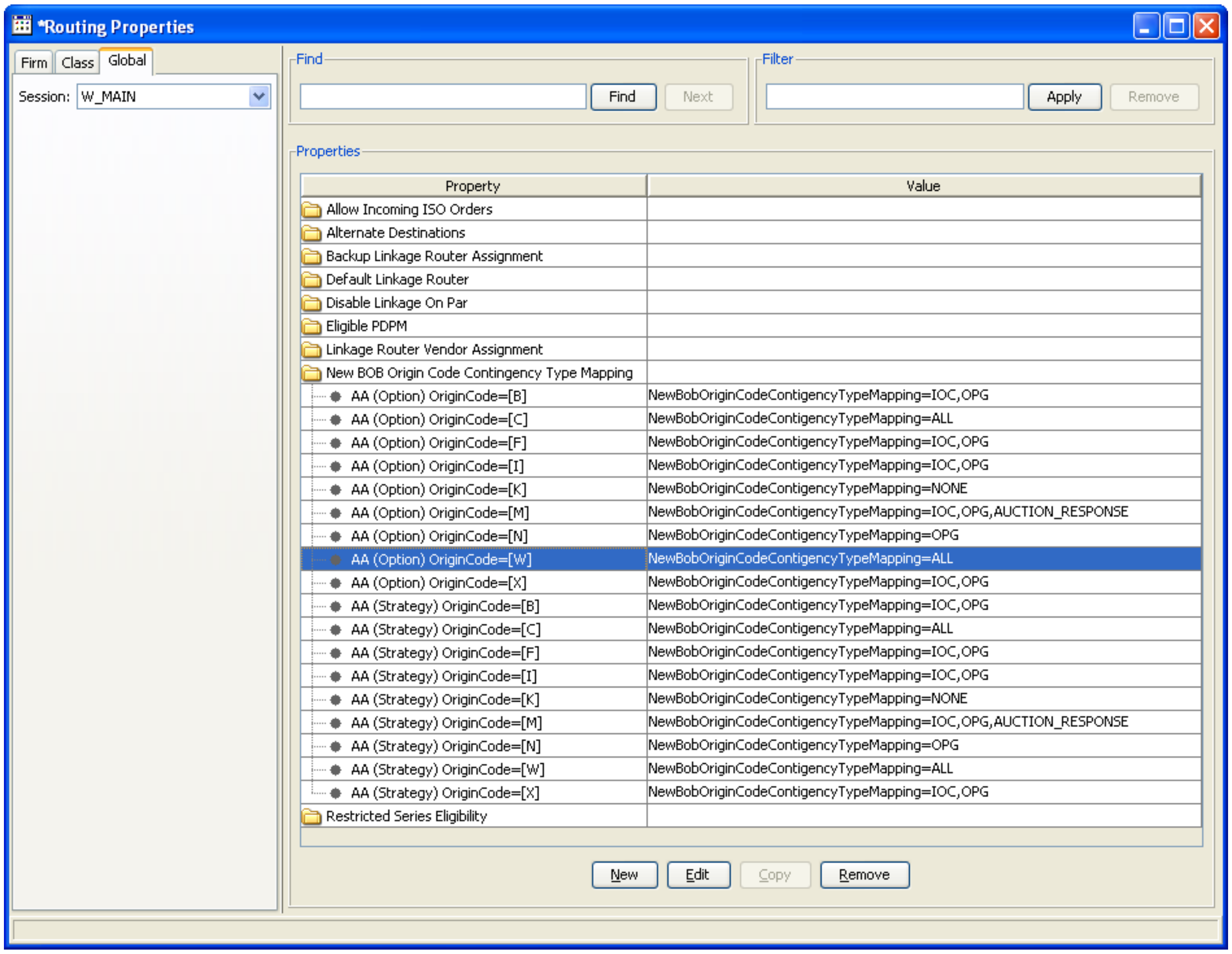
Property Type: Allowed SAL Origin Codes

Product Class	Value
Default	Allowed New SAL Origin Codes=C-CUSTOMER (3), Enabled=true; Allowed New SAL Origin Codes=W-BROKER_DEALEF

Allowed New SAL Origin Codes: C-CUSTOMER (3)

Enabled: ✓ true

Verify Contingency table to below screen shot. Using AA for class SPX



Session: W_MAIN

Properties

Property	Value
Allow Incoming ISO Orders	
Alternate Destinations	
Backup Linkage Router Assignment	
Default Linkage Router	
Disable Linkage On Par	
Eligible PDPM	
Linkage Router Vendor Assignment	
New BOB Origin Code Contingency Type Mapping	
AA (Option) OriginCode=[B]	NewBobOriginCodeContigencyTypeMapping=IOC,OPG
AA (Option) OriginCode=[C]	NewBobOriginCodeContigencyTypeMapping=ALL
AA (Option) OriginCode=[F]	NewBobOriginCodeContigencyTypeMapping=IOC,OPG
AA (Option) OriginCode=[I]	NewBobOriginCodeContigencyTypeMapping=IOC,OPG
AA (Option) OriginCode=[K]	NewBobOriginCodeContigencyTypeMapping=NONE
AA (Option) OriginCode=[M]	NewBobOriginCodeContigencyTypeMapping=IOC,OPG,AUCTION_RESPONSE
AA (Option) OriginCode=[N]	NewBobOriginCodeContigencyTypeMapping=OPG
AA (Option) OriginCode=[W]	NewBobOriginCodeContigencyTypeMapping=ALL
AA (Option) OriginCode=[X]	NewBobOriginCodeContigencyTypeMapping=IOC,OPG
AA (Strategy) OriginCode=[B]	NewBobOriginCodeContigencyTypeMapping=IOC,OPG
AA (Strategy) OriginCode=[C]	NewBobOriginCodeContigencyTypeMapping=ALL
AA (Strategy) OriginCode=[F]	NewBobOriginCodeContigencyTypeMapping=IOC,OPG
AA (Strategy) OriginCode=[I]	NewBobOriginCodeContigencyTypeMapping=IOC,OPG
AA (Strategy) OriginCode=[K]	NewBobOriginCodeContigencyTypeMapping=NONE
AA (Strategy) OriginCode=[M]	NewBobOriginCodeContigencyTypeMapping=IOC,OPG,AUCTION_RESPONSE
AA (Strategy) OriginCode=[N]	NewBobOriginCodeContigencyTypeMapping=OPG
AA (Strategy) OriginCode=[W]	NewBobOriginCodeContigencyTypeMapping=ALL
AA (Strategy) OriginCode=[X]	NewBobOriginCodeContigencyTypeMapping=IOC,OPG
Restricted Series Eligibility	

Appendix G - CAS & SA CAS Installation procedures

- Follow CAS Production Installation and CAS Component configuration documents for standard installation.
- After installing 8.6 CAS, each user id must be configured with correct rate limits for their respective sessions otherwise user would be limited to the default limits which are too low for some sessions.
- CBOEDIR_CRIT_8.6.74 has a TTE fix for Federated query for cancelAllQuotes which causes huge outliers on BC.
- CBOEDIR_CRIT_8.6.115 has as FIX for PIT 70218.

dix H - FIXCAS Installation Procedures

- Follow CAS Production Installation and CAS Component configuration documents for standard installation.
- After installing 8.6 FIXCAS, each user id must be configured with correct rate limits for their respective sessions otherwise user would be limited to the default limits which are too low for some sessions.
- CBOEDIR_CRIT_8.6.74 has a TTE fix for Federated query for cancelAllQuotes which causes huge outliers on BC.
- CBOEDIR_CRIT_8.6.115 has as FIX for PIT 70218.

Appendix I - MDCAS Installation Procedures

- Follow CAS Production Installation and CAS Component configuration documents for standard installation.

Appendix J - CFIX Installation Procedures

None. Manual installation is needed.

Appendix K - MDX Instllation Procedures

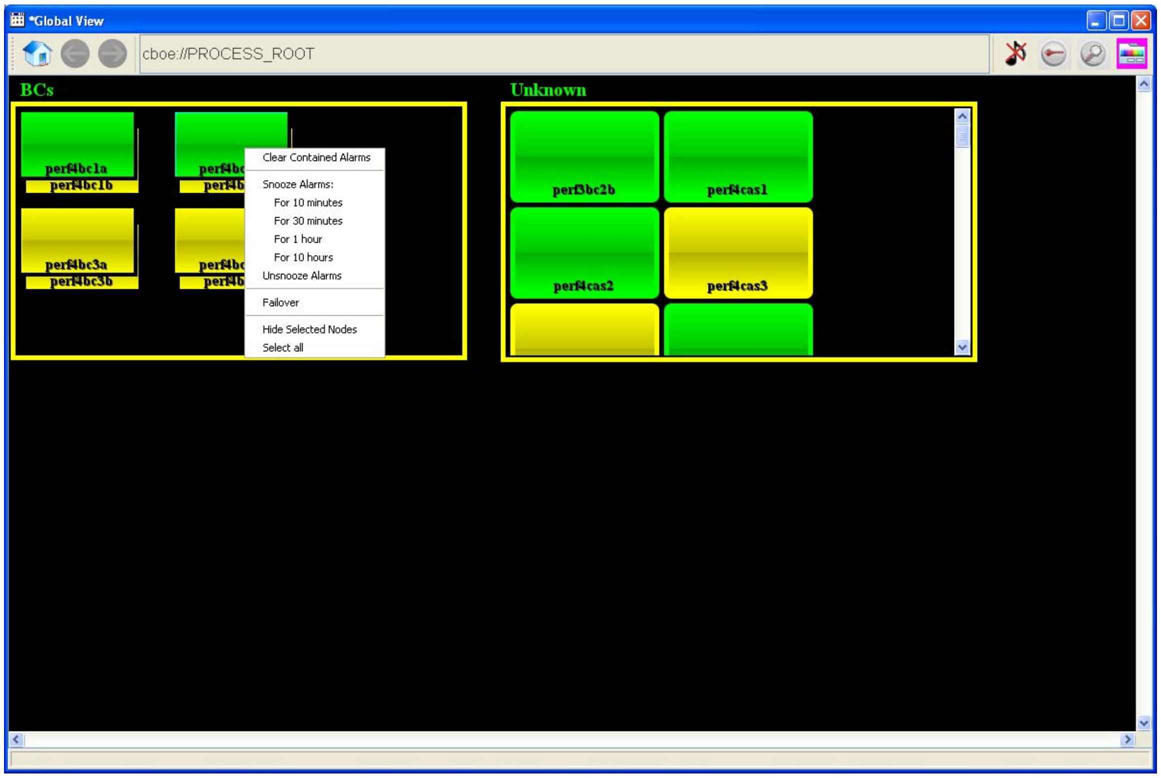
- Follow CAS Production Installation and CAS Component configuration documents for standard installation.

Appendix L - Performance tests

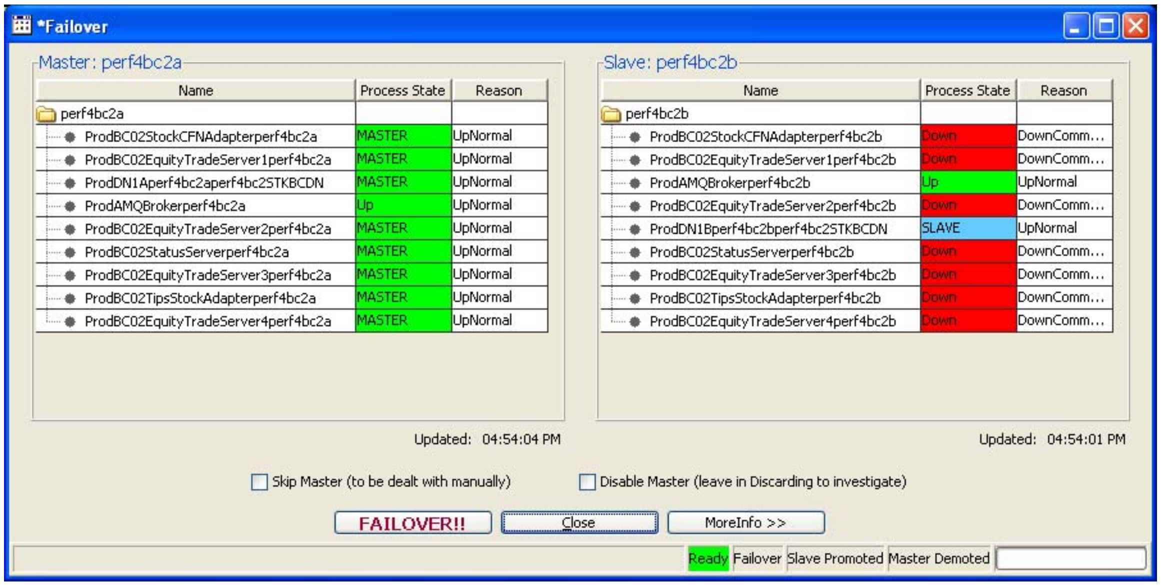
Nbr	Test	Scenario
1	Base line Test	Steady customer order/cancel with no synching
2	Basic latency timing and failover	Steady customer order/cancel and failover
3	Verify failover with huge nbr of orders	Send 1 million orders as fast as possible.
4	Impact of slave bounce	1 hour test , bounce slave in between to measure impact on master.
5	Impact of slave hangs	1 Hour test and ppstop slave for 5 minutes to measure impact of slave hangs.
6	Single and double failover	1 hour test with failover and then failover again to verify we can go back to original master.
7	Controlled synching test	Submit 100,000 initially followed by order /cancel pair, stop test and verify order counts on OHS and TS.
8	Verify master startup intraday	Submit 100,000 initially follwed by order /cancel pair, stop test , bounce master bc & verify order counts on OHS and TS.
9	Verify slave startup before master works correctly	Bring up master, send 100,000 orders, kill the master, bring up slave,bring up master and verify order counts in OHS and TS on master and slave.
10	Data Integrity verification	Restart Master while trafic is running
11	Data Integrity verification	Custome test to print Master order count and kill with system.exit

Appendix M – Fast Failover GUI

In order to provide a more user-friendly failover experience the Global View GUI has been enhanced to provide visual feedback to the Fast Failover process. The Fast Failover Window is invoked by clicking on a cluser in the Global View Screen as follows:



Choosing the Failover option the user will show the following Failover window:


*Failover

Master: perf4bc2a

Name	Process State	Reason
perf4bc2a		
ProdBC02StockCFNAdapterperf4bc2a	MASTER	UpNormal
ProdBC02EquityTradeServer1perf4bc2a	MASTER	UpNormal
ProdDN1Aperf4bc2aperf4bc2STKBCDN	MASTER	UpNormal
ProdAMQBrokerperf4bc2a	Up	UpNormal
ProdBC02EquityTradeServer2perf4bc2a	MASTER	UpNormal
ProdBC02StatusServerperf4bc2a	MASTER	UpNormal
ProdBC02EquityTradeServer3perf4bc2a	MASTER	UpNormal
ProdBC02TipsStockAdapterperf4bc2a	MASTER	UpNormal
ProdBC02EquityTradeServer4perf4bc2a	MASTER	UpNormal

Updated: 04:54:04 PM

Slave: perf4bc2b

Name	Process State	Reason
perf4bc2b		
ProdBC02StockCFNAdapterperf4bc2b	Down	DownComm...
ProdBC02EquityTradeServer1perf4bc2b	Down	DownComm...
ProdAMQBrokerperf4bc2b	Up	UpNormal
ProdBC02EquityTradeServer2perf4bc2b	Down	DownComm...
ProdDN1Bperf4bc2bperf4bc2STKBCDN	SLAVE	UpNormal
ProdBC02StatusServerperf4bc2b	Down	DownComm...
ProdBC02EquityTradeServer3perf4bc2b	Down	DownComm...
ProdBC02TipsStockAdapterperf4bc2b	Down	DownComm...
ProdBC02EquityTradeServer4perf4bc2b	Down	DownComm...

Updated: 04:54:01 PM

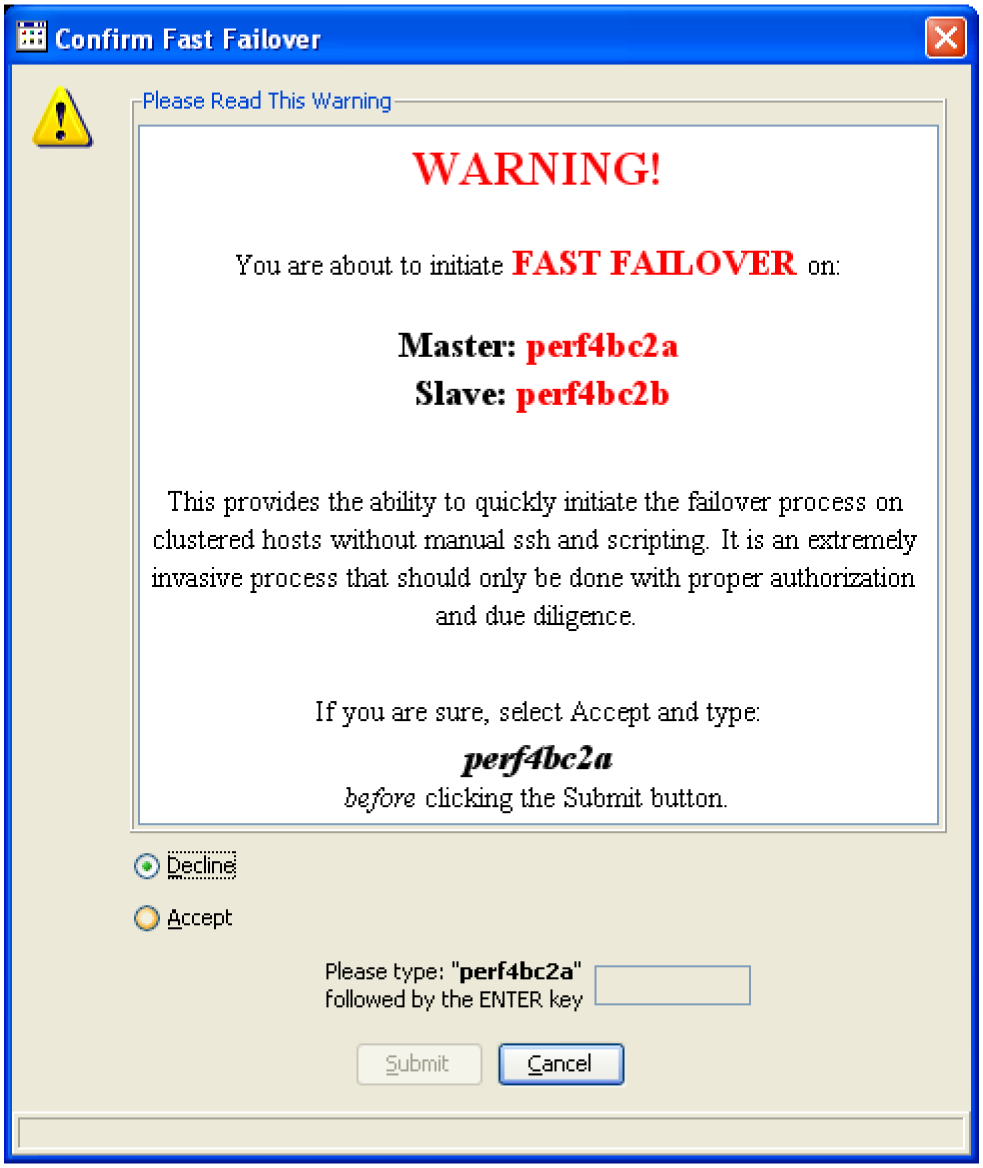
☐ Skip Master (to be dealt with manually)   ☐ Disable Master (leave in Discarding to investigate)

FAILOVER!! | Close | MoreInfo >>

Ready | Failover | Slave Promoted | Master Demoted

The GUI calls the same code that the command line version calls to do the Failover. This will autodetect which machine is master and display it on the left and the slave on the right. Clicking the "FAILOVER!!" button will start the failover process. The bottom status line follow the Failover status and highlight the Failover, Slave Promoted, and Master Demoted indicators as the failover progresses. It will also display the failover time in milliseconds. Also, the process states on both boxes are depicted just like they are in the System Health Monitor and will reflect process states as they are published by Process Watcher. This may not happen immediately and might take as long as 10 seconds to update.

In order to provide a safeguard to make sure that the user is attempting to failover the proper BC, they will have to type the BC name into the following warning dialog after initiating the failover:



Confirm Fast Failover

Please Read This Warning

WARNING!

You are about to initiate **FAST FAILOVER** on:

Master: perf4bc2a
Slave: perf4bc2b

This provides the ability to quickly initiate the failover process on clustered hosts without manual ssh and scripting. It is an extremely invasive process that should only be done with proper authorization and due diligence.

If you are sure, select Accept and type:
perf4bc2a
before clicking the Submit button.

[✓ Decline]
Accept

Please type: "**perf4bc2a**"
followed by the ENTER key

Full documentation of this new GUI window can be found in the System Health Monitor Users Manual.