CBOED*irect* Release Notes

CboeD*irect* 8.7

Last updated: July 27, 2011

Authors: Steven Sinclair

Table of Contents

Overview of CD Server changes in this release

Work requests included in this release

- 6800 – Trade Latency Enhancement
- 6836 – Optimize Order Business Logic
- 6745 – Spread Protections (Phase 1)
- 6834 – Order History Optimization
- INFRA 14.1 WRs & SEDLs
  6843 – CORBA SDP Transport
  6851 – cmdConsUtil ssh / localhost bugfix
  6855 – OutlierMonitor M/S Negotiation, IGC PW channel to CAS/FIX, "stats" scripts in startInfraBase
  6868 – Active MQ Version 5.4.2, IIOP Socket Proxy lock bugfix
  PITS/SEDL 63285 – PW Auto-Registration bugfix (excessive logging)
- 6969 – (interim) Single-stock circuit breakers for CBSX
- 6973 – (interim) COB/COA/AIM configuration for C2 (a.k.a. "SPX PM for C2")
- 6959 – (interim) Linkage fill report to include executing exchange

WR 6800 : Trade Latency Enhancement

Goal

The purpose of the project feature is to improve the overall trade server performance, per trade execution time and overall trade volume, by performing the trade activities asynchronously (i.e. moving the activities to an off-thread) within the trade server.

The trade activities that are being made asynchronous are:

- trade report(s) creation(package, atomic and child level)
- fill reports generation for the involved tradable(s)
- trade report(s) publishing
- fill reports publishing
- last sale(i.e. ticker) publishing

Overall the feature is business neutral as no new requirements are implemented and no existing business use case is impacted.

Description

During the order or quote (i.e. tradable) execution, the trade server currently spends significant of time within the class-level lock for performing the trade related activities after the allocating the trade. Apart form the updating of tradable(s) for the allocated quantity and order book for the involved tradable(s) positions, the remaining trade activities (i.e. operations listed above) are basically to record trade(s) for book keeping purposes and publish the trade(s) for confirmation and regulatory purposes.

The trade record creation and persistence does not have to be carried out within the class-level lock and the trade publishing is already asynchronous in nature as it is published via event channel(s), so to achieve the stated purpose, the project effort basically moves the tradable and order book updates to main-thread and performs the remaining trade activities in the asynchronous thread.

The project introduces configurable new asynchronous thread-pool and to guarantee the processing sequence, the new asynchronous trade commands are allocated to threads based on the product key hashing. And, to avoid any rollbacks due to pre-mature order or quote execution or to complete the atomic execution (e.g. multi price points), the asynchronous trade commands execution is only enabled upon the successful completion of root- transaction in the main-thread.

To maintain and retain the existing trade server behavior, either functional or process related, the dependent operations within the trade server are synchronized with the pending asynchronous trade(s) completion. For example, the cancel fills of an immediate or simultaneous cancel following the tradable execution or STS leg fills as a result of trading with HTS/ETS are synchronized for any pending asynchronous trade(s) completion.

Though the overall the project is business neutral, one known side effect with asynchronous trading comparing with current system is that the CM is published before the last sale so any internal or external client (e.g. ITG Client or client Firm) is expected to be aware of and work with the new publishing sequence.

WR 6745 : Spread Protection Phase - I

Goal

The goal of the CBOEDIR_8.7 Release Notes project is to stop spread executions against market maker quotes in Spread Protection enabled classes.

Description

Currently orders resting in COB will trade when they are hit by an incoming spread, re-COA'd or when the legs line up. Since spreads will trade when the legs line up and they trade at the leg prices, there is a possibility that they will trade at erroneous prices. Further, some users put spreads in the book that are intended to only trade if the legs are priced incorrectly. For example, a vertical 100-110 call spread can never be worth more that $10, however someone could put an order in COB to sell at $10.01. That order could then trade at $10.01 or an even higher price if a market maker accidentally enters a bad quote. This first phase addressed in this feature will prevent spread executions at erroneous prices.

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 Trading Property, Routing and Cancel Reason

1. Allowed Complex Trades With Quotes – *Trading Property*

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

This property is used to enable Spread Protection feature. If this property is set to false, Spread Protection feature is applied to the spread orders. The default value is true, that means those products are not spread protection enabled

2. ORDER_TOUCHES_LEG_QUOTES (5457)– New Routing Reason

This is the new routing reason used when the spread order becomes marketable with leg quotes, and that order is sent to PAR with this reason code.

3. ORDER_TOUCHES_LEG_QUOTES – New Cancel Reason

This is the new cancel reason used when the spread order becomes marketable with leg quotes, and that order is cancelled in C2. Also in the case of Spread "I" order when the order becomes marketable with the leg quotes, that order will be cancelled with this reason.

How to turn on "Spread Protection – Pase I" functionality

1. Verify the trading property"Allowed Complex Trades With Quotes " is set to false for option as well as Strategy class. The below should return false.

setTradingProperty W_MAIN <class symbol> option AllowComplexTradesWithQuotes false

setTradingProperty W_MAIN < strategy symbol> strategy AllowComplexTradesWithQuotes false

How to turn off “Spread Protection – Pase I” functionality

1. Set the trading property “Allowed Complex Trades With Quotes ” for the option as well as Strategy class to true. The below should return true

setTradingProperty W_MAIN <class symbol> option AllowComplexTradesWithQuotes true

setTradingProperty W_MAIN < strategy symbol> strategy AllowComplexTradesWithQuotes true

WR6836: Optimize Order Business Logic

Goal

Review code path for order execution in the OHS and trade engine (TE) processes, and make incremental performance improvements. Performance tests are expected to reflect a shorter call duration in both OHS and TE.

Description

Order Business Logic Optimizations – Trade Engine

1. Refactor SweepEvaluationService to not reget HAL info in the execution path if not required.
    a. HALHelper
2. OrderBookImpl -> checkAndCancelSameSideManualQuote(…)
    a. Invoked for bob classes as well today, make change to only invoke for non-bob
3. BrokerProcessorBase-> checkBookedOrderWithLockCheck(…)
    a. There is a call to iterate all tradables from book for MKT order that hasn't traded… This is huge, Modify OrderbookImpl.getTradables(…) to skip all the tradables for MKT orders.
4. BrokerProcessorHybrid2Impl ->
    a. doTradeAndSweepOrderProcessing : This method has called getContingencyType method several times and gets same result each time, since contingency type is not going to change while order flow is in progress. We can have switch statement to get contingency once and check for nbbo protection. GetContingencyType is an expensive call
    b. preProcessSweepTradable - remove markFillType. We have emit point in processInternalSweep (do we need this emit point) - Only thing we see is getContingencyType called 2 times
    c. processHALTradable : getSweepHALInformation call can be remove immediately because this call is just setting the halFlashPrice and which is not required in internal sweep processing.
5. BrokerProcessHybridImpl->
    a. autoExProcessing : getOrderBook should get called inside the index hybrid check, so it will not get executed in regular order path where it is not required.
    b. BrokerProcessorHybridImpl->regularOrderProcessing : findActiveAuction method called twice, processPotentialEndOfAuction method should get called only if auction is active. if we move this call inside if statement then we can save second call
    c. isHALLikeAuctionGoingOn : In doAuctionHALProcessing method, isHALLikeAuctionGoingOn method called twice. it returns Boolean variable which can be used next time instead of calling method again.

6. BAHALProcessingStrategy->handleHALOrderAuction : OrderBook.refreshBestBook is called twice, First inside the processHALTradable and immediately after the method call. This needs to be discussed further.
7. OrderImpl->getSide() :
    a. This method is called 52 times in acceptOrder path (from call tree), and has an if-else-if structure. Can be replaced by switch-case, or can have only one object of the side.
    b. needsNBBOProtection() : This method has called getContingencyType method several times and gets same result each time, since contingency type is not going to change while order flow is in progress. We can have switch statement to get contingency once and check for nbbo protection. GetContingencyType is an expensive call
    c. getQuantityAllowed() : Method getRemainingQuantityForTrade is called twice, once from inside if clause and again while method returns. One call can be avoided. Also, getShipQuantity method is called multiple times from getRemainingQuantityForTrade. This quantity can be cached instead and re-used.
    d. getAutoLinkOriginCodes() : getAutoLinkOriginCodes method can be changed to return hashmap instead of array of auto link origin codes, and get the value from map instead of iterating all the sorted origin code where W is always at the end
    e. getContingencyType() -> Ensure no redundant calls are being made to this method, can be retrieved once and used
    f. getRemainingQuantity() -> Ensures no redundant calls are being made to this method
    g. getremainingQtyForTrade().getRemainderHandalingMode called twice, can reduce to 1
8. AuctionProcessorBase ->
    a. getAuctionMinQuoters : Instead of looking through the trading product home, getting the trading class impl and fetching the trading property, we can pass trading class from broker to get the trading property
    b. AuctionProcessorBase-> getAuctionParticipants :getAuctionReceiver type can be get it directly from the Treading class when we get auction min quoters, no need to llop again to find trading product and then trading class
    c. getAuctionParticipants : getQuotes method returning quoteImpl array and in further processing user id and key is retrieved from the quoteImpl, instead of returning quote impl can be returned only user key, which will satisfy the requirement. So we will save two iterations for all quotes
9. isHALEligible() -> Need to change trading property to return map for AutoLinkOriginCodes instead of Array, the originCodes also can be returned as a Map instead of array
10. OrderBookSideItem->A new variable of Object type Side was added and is returned on a get, instead of comparing what side the order is on, and then return the Side object. On Set we replaced multiple if-else statements with switch statement.
11. OrderBookPriceItem.getInstance() -> Initialize method can be removed as when object get checkout it has all the values initialize by default, we don't have to set it again.
12. OrderBookPriceDetailImpl-> reset() -> reset method we found we do a getTradable(), there were 2 fields tradableType and tradableKey, which is not required to lookup an

order or a quote tradable object, the field detailedTradable object is already set for this price detail, which can be returned instead.

13. OrderRoutingDestinationBaseImpl-> isDirectedAIMOrderReturned() -> check Optional data fierst and then check extension and third should be extensionsFieldParser.
14. ORDI.acceptOrder() ->
    a. isValidAIMIOrdersForBobClass and setHandlingInstructionForBOB method can be restricted to BOB class only.
    b. isDirectedAIMPrimaryOrder() ->anOrder.isInternalizedOrder() check move first and then do the extensionHelper parsing.
15. BrokerStrategyFactoryHybridImpl->
    a. findHALStrategy -> exchange getAuctionEnabled condition in if statement, because most of the new hal is enabled, getOrderBook call is removed as it was never used anywhere
    b. isProductStateValidForHALAuction -> TradingProduct is passed instead of looking one more time in map
16. QuoteRiskManager -> takeQuoteRiskAction -> clear hashMap only when the list is not empty -> AcceptAsyncCancelQuoteCommand BrokerImpl UnderlyingMarketDataServiceImpl UserDefinitionServiceImpl AlertAgentSerivceHomeImpl DerivedQuoteSideMap UserInputMonitorImpl UserPreferenceCahceMap UserServiceCacheManager UserInputMonitorImpl UserInputMonitorSegment ExchangeSerivceCacheManager UserPreferenceCacheManager UserServiceCacheManager
17. Handling Instruction refactoring to replace stringTokenizer and do ordinal indexing for improvement and use reflection free
18. There is performance overhead by calling these two methods on the same Hashtable,containsKey() and get(). Both these methods do the same thing, except, containsKey() returns true/false and get()returns the actual Element. If the Hashtable is of considerable size then the overhead is high -> BrokerHomeImpl.findByProduct(.), MarketDataReportProcessorImpl.getRecapForProductFromCache(), QuoteHomeImpl,OrderValidationOpenOutcryStrategy,ReCOAManagerImpl, ReCOAMonitorungServiceImpl, AuctionValidationStrategyBaseImpl, LinkageClassGateHomeImpl

Order Business Logic Optimizations – OHS

1. OrderHistoryImpl.isZeroOrNoPrice(…), user primitive comparison instead of object.
2. WFStepCheckRoute – Retrieve extensions from Helper only if is not null in Order
3. AbastractBaseProperyKey – Initialize HashMap in the set method, so that it is not initialized for every property created
4. WFStepHandlingInstruction –
5. Removed creation of an extra HandlingInstruction object in the beginning and left it to be initialized in specific conditions.
6. Move string buffer creation inside if condition and
7. ReasonabilityEditServiceImpl
8. Added a zero-price static field.
9. isOrderSuperMarketable(…) moved getQQV3 call after MKT price check.
10. isZeroOrNoPrice(…) -> Use primitive comparison instead of object comparison.

11. WFStepRejectOnProductState
12. Moved StringBuffer creation to if conditions for audit history creations.
13. WFStepAppendPDPMDataForDAIM
14. Check DAIMness of the order only if the order is of internalized type.
15. WFStepCheckIndexHybridReserveOrder -> Moved QQV3 call inside the if.
16. PriceFactory, fixed ClassCastException caused in createPriceSqlType when cacehPrice flag was turned on in OHS.
17. OrderHandlingInstructionImpl – Changes to replace StringTokenizer with an ordinal comparison.
18. OrderHandlingInstructionNoReflectionImpl - Changes to replace StringTokenizer with an ordinal comparison.

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: 6969 – (interim) Single-stock circuit breakers for CBSX

Goal

In order to allow for the expansion of the single-stock circuit breaker pilot, the away market quote feed local to a BC (i.e., the StockTipsAdapter) should invoke the HALT of the CBSX products. Currently, this is executed via the global server in conjunction with the global TipsAdapter, which is not expected to be performant or easy to manage as the number of symbols eligible for this type of handling expands.

Summary of changes

Modified StockTipsAdapter code so that when a 'halt' indicator is received by the process from the stock XTP feed, a state change to CBSX HALT is invoked for the same equity or index.

WR: 6973 – (interim) COB/COA/AIM configuration for C2

Goal

The CBOE2 exchange needs to be able to support spread products, a complex spread book, and AIM auctions in order to properly support SPX derivatives.

Summary of changes

Configuration and minor BC code changes to support a spread trade server and related functionality in C2 environments

WR: 6959 – (interim) Linkage fill to include executing exchange

Goal

Firms need to be able to know which exchange filled any given order volume.

Summary of changes

Changes made in OHS and CAS to support the propagation of the executing exchange for linkage fill reports to the firm.

WR: 6996 – (interim) MDX 8.7 interim

Goal

Update MDX to new INFRA, add new quote conditions, callback removal log enhancement

Summary of changes

1. INFRA update: new MDX config spec point to CBOEDIR_CRIT_8.7.71, which refer to QAINFRA_CRIT_14.1.19, code change made for startup dependency.
2. Two new CMI constants added for quote conditions: FirmBidNonFirmAsk and NonFirmBidFirmAsk.

In cmiConstants.idl:

```
interface MarketIndicators
  {
  …
       const cmiMarketData::MarketIndicator FIRM_BID_NON_FIRM_ASK    = 36;
       const cmiMarketData::MarketIndicator NON_FIRM_BID_FIRM_ASK    = 37;
    };
```

3. Add user name in the error message passes to CAS, so that CAS knows the source of call back removal.
4. Log enhancement :

There is a new line in config/bin/setContext:

export CLIENT_DEBUG_APP=On // turn on debug log

or: export CLIENT_DEBUG_APP=Off // turn off debug log

5. NBBO calculation fix: PITS # 58461, Server have it fixed, new test plan added. Refer to document: PITS_58461_TestPlan.

Bugfixes included in this release

SEDL Tickets Merged In Release

1. SEDL#8214 – STOP order triggered on a LATE price report
2. SEDL#8805 – AIM responses should be rejected
3. SEDL#8810 – Manual price cancel to tape not working in some cases in SPX
4. SEDL#8834– Some AIM auction ending early but reported in history as 'normal ending'
5. SEDL#8866 – Inbound ISO orders not executing when linkage is disabled.
6. SEDL#8877 – Timing issue on partial cxl/re [Test Plan n/a]
7. SEDL#8909 – Stale quote result returned when a runtime exception is in quote validation [Test Plan n/a]
8. SEDL#8922 – OHS routing failure for OMT order should keep the order on the OMT.
9. SEDL#9175 – TipsAdapter script post-failover logs errors (deprecated script) [Test Plan n/a]
10. SEDL#9264 – TipsAdapter in generated process watcher list is incorrect. [Test Plan n/a]
11. SEDL#9297 – AON quote disseminated to OPRA (need new CM view for OPRA)
12. SEDL#9304 – Spread order trade at leg through resting AON price
13. SEDL#9319 – Duplicate order book callbacks after a session restart event causes errors [Test Plan n/a]
14. SEDL#9337 – Deprecate 'bad prod key' alarm log to be info. [Test Plan n/a]
15. SEDL#9374 – [SAGUI] logout issue
16. SEDL#9443 – Linkage fill reports queuing in OHS – need to multithread reports [Test Plan n/a]
17. SEDL#9521 – Linkage fill report not matched with any original order [Test Plan n/a]
18. SEDL#9534 – Evaluate all PAR trades for firm quote compliance
19. SEDL#9536 – Manual price reporting not working inC2
20. SEDL#9539 – CBSX orders booked in pre-open causing quotes to disseminate
21. SEDL#9559 – deleteStrategiesWithNoOrders should trigger email to support folder [Test Plan n/a]
22. SEDL#9597 – [SAGUI] NullPointer when disabling a user group
23. SEDL#9636 – [CAS] Slow fill report delivery from CAS to FIRM [Test Plan n/a]
24. SEDL#9638 – Interceptor alarms when class name is not found (annoyance in log files) [Test Plan n/a]
25. SEDL#9639 – Redundant alarms for InvocationTargetException clogging log files. [Test Plan n/a]
26. SEDL#9653 – AIM Mkt-to-Mkt spread trades .02 better rather than .01 better.

27. SEDL#9655 – Log which class(es) are causes RTT messages to occur [Test Plan n/a] (improve logging)
28. SEDL#9657 - $0 COB auction reponses are rejected
29. SEDL#9702 – maxThreadPoolSize trading property exception getting excessively logged [Test Plan n/a]
30. SEDL#9703 – Triggered STP order overfills after auction [Test Plan n/a]
31. SEDL#9719 – Order hist. entries with 0 time (add safety code to alarm & use curr time) [Test Plan n/a]
32. SEDL#9746 – VIX series did not open due to $0 trade price validation error
33. SEDL#9750 – Update setEnv to reference infra classpath config rather than infra jars [Test Plan n/a]
34. SEDL#9764 – Class params not initialized until first spread intraday add [Test Plan n/a]
35. SEDL#9765 – Spread ordering is by month name instead by number #
36. SEDL#9766 – Book Depth Incorrect
37. SEDL#9772 – Add jmap command to shutdownTradeeng script [Test Plan n/a – verified via perf rollout]
38. SEDL#9776 – GTC cxl/re can cause the cancelled order to be re-instated on next day
39. SEDL#9799 – Migrate to from JDK 1.6.0_22 to 1.6.0_23. [Test Plan n/a]
40. SEDL#9805 – Change JVM flags to for some processes as identified in ticket details. [Test Plan n/a]
41. SEDL#9812 – Class cancel optimization for redundant cancels [Test Plan n/a]
42. SEDL#9813 – Enable standard emit points on STS [Test Plan n/a]
43. SEDL#9832 – MarketBuffer consumers to be protected from 0-length field messages [Test Plan n/a]
44. SEDL#9835 – COA Midpoint
45. SEDL#9836 – Class cancel pre-processing enhancement [Test Plan n/a]
46. SEDL#9845 – CM Bridge ONE/CFE/COF event batching [Test Plan n/a]
47. SEDL#9853 – Empty JGrinder commits (avoid them) [Test Plan n/a]
48. SEDL#9857 – Correct GlobalServer routing proxy to avoid potential failover session-state issues [Test Plan n/a]

Operator Procedure changes

1. None

Infra 14.1 Changes

Summary of changes

- *DN Refactoring:* The DN code was heavily refactored for reliability and ease of maintenance. The most noticeable change is tolerance for and recovery from situations where a dependent AMQBroker is down at DN startup time. Currently the DN initialization "hangs" waiting for all AMQBrokers to be available.
- *ActiveMQ Version 5.4.2*: Upgrade of the AMQ code from version 5.2 to the latest "gold" release. This will resolve open SEDL 9943
- *Messaging Transport Refactoring*: The base transport code right now is tied tightly to the concrete IIOP and TIOP implementations and has little room for change. This work will redesign the transport class structure, allowing for "pluggable" concrete transport implementations. This will be the basis for adding new transports supporting SDP, Infiniband, and Fast IPC.
- *ORB Layer –D standardization*: A generalized ORB Property handling package will consolidate and standardize ORB tier –D switches. "Legacy" –D switches will not be affected, however, relative to the end-user applications.
- *IIOP Socket Code Refactoring*: The current socket-layer code implementation is showing its age and is becoming extremely difficult to maintain. The base socket interfaces will be redesigned and recoded.
- *Maintenance*: Several code "clean-up" tasks were completed to eliminate obsolete and/or deprecated code. Other maintenance tasks were done to eliminate known errors.
- *Filter metrics:* A count instrumentor was added to the filter implementation to provide a filter evaluation metric.
- *InfraEnv:* Rewrite of InfraEnv in perl. MD5 digest meta-data file support for daily file version verification. Eliminate need for InfraEnvLinux.
- *DynamicConnection/ThreadPool:* Support, via FF interface, capability for the CAS to dynamically change the POA thread pool size and usage, as well as the IIOP connection pools.
- *Send BC PW updates to CAS-layer:* Update ExtentMap to handle sending out PW updates to the CAS layer for BC processes.
- *InfraBase:* Added timed_nicstat, timed_vmstat and timed_mpstat to InfraBase processes.

Appendix A - Database Changes

GC Database changes

There are no GC Table changes at this time.

BC Database changes

The following index should be created for all BC database schemas to improve the responsiveness to user order query operations:

```
CREATE INDEX SBTORDER_I6 ON SBTORDER (PRODUCTKEY,USERID) LOCAL;
```

Appendix B – Installation order and pre-requisites

Standard installation sequence. No special tier dependencies.

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 for enabling Strategies and Auction only when SPX is suppose to go live.
- ☐ Important: 8.7 No longer has different ACLs for C1 and C2. Please use ACL.csv (C2ACL.csv is deprecated) everywhere.

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

   execute "$RUN_DIR/bin/addAllowComplexTradesWithQuotes ". This will set the default trading property for Allow Complex Trades With Quotes.

   Execute "getTradingProperty W_MAIN default AllowComplexTradesWithQuotes" and verify that output is "AllowComplexTradesWithQuotes = true"

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). **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. Add SpreadTradeServer1 to ALL_SERVER_NAMES in setContext
4. For SPXPM on C2 BC install, where SPXPM class has been created
    a) Add export EMPTY_ORIGIN_FOR_COMPLEX_NON_CUST=true to setContext.
5. Shutdown tradeengine on the BC being installed (tradengp and tradengh login).
6. Install new software CBOEDIR.xxx release, change run_dir links .
7. Start the BC processes (tradengh first and then tradengp logins).
8. Enable business external connections on the master bc.
9. 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

i. `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	Enter Manual Price Report – Verify Last Sale Update – Verify MDH entry	
18	**<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.	
19	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 G – CAS Installation procedures

CAS Rollout steps

- 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.
- NOTE : With SPXpm BC rollout – New ACL should be installed on the CASes.
- Please Read "GC2 A/B Trade engine Installation Procedures" for new 8.7 ACL for C2.
- During the 8.7 **CAS Test Environment Rollout (ATG, PERF, API, DEV, and ALL NON-PROD environments)** , Add the following exports to each **CAS' setContext.v2cas01**
    - export USE_MANAGED_THREADPOOLS=true
    - export USE_MANAGED_CONNECTIONPOOLS=true
    - export DYNAMIC_CONNECTIONPOOL_GROWTH=true
    - export DYNAMIC_THREADPOOL_GROWTH=true
    - export USE_USERLOADMANAGER_STARTTIMES=**false**

  Important Note-1: These exports are for **CAS ONLY**. (i.e. **Not needed** for FixCAS, SACAS, MDCAS)
- During the 8.7 CAS **Production Rollout**, Add the following exports to each **CAS' setContext.v2cas01**
    - export USE_MANAGED_THREADPOOLS=false
    - export USE_MANAGED_CONNECTIONPOOLS=false
    - export DYNAMIC_CONNECTIONPOOL_GROWTH=false
    - export DYNAMIC_THREADPOOL_GROWTH=false
    - export USE_USERLOADMANAGER_STARTTIMES=false

  Important Note-1: These exports are for **CAS ONLY**. (i.e. **Not needed** for FixCAS, SACAS, MDCAS)

  Important Note-2: We will rollout with exports exactly as set above. However, at some point after or during the production rollout of 8.7 CAS we need to set all flags to true to activate UserLoadManager functionality.

Appendix 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.

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 Installation Procedures

- Follow MDX Production Installation and MDX Component configuration documents for standard installation. New change is:

1. There is a new line in config/bin/setContext:

   export CLIENT_DEBUG_APP=Off