CONFIDENTIAL

CBOE Application Programming Interface

CBOE API Version 5.0 - Release Notes

Provides an overview of upcoming changes in the next production release of the CMi

CBOE PROPRIETARY INFORMATION

07 March 2008

Document #[API-00]

Front Matter

Disclaimer

Support and Questions Regarding This Document

Questions regarding this document can be directed to The Chicago Board Options Exchange at 312.786.7300 or via e-mail: api@cboe.com.

The latest version of this document can be found at the CBOE web site: http://systems.cboe.com/webAPI.

Table of Contents

FRONT MATTER I

DISCLAIMER I
SUPPORT AND QUESTIONS REGARDING THIS DOCUMENT I

TABLE OF CONTENTS 2

OVERVIEW 3

CMI V5.0 HIGHLIGHTS 3

AIM AUCTION FOR COMPLEX ORDERS 3
ADDITIONAL BEHAVIOR IN TOO LATE TO CANCEL SCENARIOS 4
AIM SOLICITATION MECHANISM (AIM AON) 4
RETRIEVING THE CURRENT RATE LIMITS 4
NEW QUOTE CANCEL METHODS 5

IDL INTERFACES 6

DOCUMENT CHANGES 16

API-01 16
API-02 16
API-03 16
API-04 16
API-05 16
API-06 16
API-07 17
API-08 17
CAS-01 17
CAS-02 17

SIMULATOR 17

TEST PLAN CHANGES 17

Overview

Release notes for the CMi API, Version 5.0, were initially published on February 29, 2008. Version 5.0 supports new CMi V5 interfaces, documentation changes and a new Simulator. Since the initial publication, details have been added to this document for the AIM Solicitation Mechanism (AIM AON). Additionally, on April 24, 2008, the documentation was updated to include additional functionality for Too Late To Cancel inactive orders (Express orders).

The sections below detail the changes in this release. Your feedback or questions regarding this document should be sent to api@cboe.com.

CMi V5.0 Highlights

The CMi V5 interfaces support three new features: (1) the ability to participate in complex order auctions, (2) the ability to query the current rate settings for a given session and (3) a new way of cancelling quotes that allows the user to suppress the corresponding cancel reports.

AIM Auction for Complex Orders

CMi users can participate in a complex order auction using the new method, *acceptInternalizationStrategyOrder*. This method has been added to the *OrderEntry* interface. The new method takes two additional parameters, *primaryOrderLegEntries* and *matchOrderLegEntries*. Similar to the existing method *acceptInternalizationOrder* used for entering AIM auctions for regular orders, the new parameters specify the leg order entry structs for the primary and matching order.

```
interface OrderEntry : cmiV3::OrderEntry
{
   cmiOrder::InternalizationOrderResultStruct
      acceptInternalizationStrategyOrder(
         in cmiOrder::OrderEntryStruct primaryOrder,
         in cmiOrder::LegOrderEntryStructSequence
primaryOrderLegEntries,
         in cmiOrder::OrderEntryStruct matchOrder,
         in cmiOrder::LegOrderEntryStructSequence
matchOrderLegEntries,
         in cmiOrder::MatchType matchType )
      raises(
         exceptions::SystemException,
         exceptions::CommunicationException,
         exceptions::AuthorizationException,
         exceptions::DataValidationException,
         exceptions::NotAcceptedException,
         exceptions::TransactionFailedException
```

```
        );
```

Additional Behavior in Too Late to Cancel Scenarios

CBOE currently tracks all orders. In order to improve order flow, CBOE will no longer store inactive orders (i.e. orders with no remaining working quantity, “Express orders”) nor process requests regarding any inactive order. Once an order has been filled or cancelled it will be removed from the active order process after a short duration. Currently, the minimum time an order could be referenced after being filled or cancelled is 30 seconds. No filled or cancelled order will exist for more than 1 minute.

If no order is found a DataValidationException with the error code INVALID_ORDER_ID is thrown. If an order exists but its remaining quantity is zero a DataValidationException with the error code NO_WORKING_ORDER will be thrown instead.

CMI users should no longer expect to always receive a TLTC messages for cancel requests as the window for their creation has been greatly reduced.

The cancel reports for inactive orders (Express orders) are no longer GMD. If a user logs out before receiving one or more Express Order cancel report, they will not be re-published as poss-resend when the user logs in again.

There is an important consequence to be noted with regard to order entry. If the original order has not been accepted by the system, the cancel request for that order will be rejected with INVALID_ORDER_ID. So, it is important to wait for the order call to return before issuing a cancel request. Since an Express Order will be cancelled or filled, they do not warrant any cancel requests. Any cancel replace request on Express orders will be rejected.

AIM Solicitation Mechanism (AIM AON)

AIM AON allows agents to electronically execute orders they represent against solicited orders.

The mechanics for entering orders into AIM AON is the same as for the existing AIM process with two main differences; 1) Both orders entered must have the contingency AON. 2) The orders must have a contract size of at least 500.

In AIM AON, the agency order will trade with the solicited order at the proposed price unless there are auction responses that improve the price of the auction for the total size of the agency order.

This process is available for simple, complex and cross product orders.

Retrieving the current rate limits

The *UserTradingParameters* interface was extended to include a new method, *getUserRateSettings*. This method takes a session name and returns the current values of various rate limits. Each rate limit is returned as a generic struct containing a key and a value, both of which are string types. Currently, this method returns the quote rate limit, quote call limit, order rate limit, order call limit and book depth call limit.

```
interface UserTradingParameters : cmi::UserTradingParameters
{
```

```
    cmiUtil::KeyValueStructSequence
       getUserRateSettings(in string sessionName)
          raises(
                exceptions::SystemException,
                exceptions::CommunicationException,
                exceptions::AuthorizationException,
                exceptions::DataValidationException
          );
};
```

New Quote Cancel methods

The existing quote cancel methods: *cancelQuote*, *cancelQuoteByClass* and *cancelAllQuotes* result in asynchronous quote cancel reports for all the quotes canceled. New aspects of the above quote cancel methods have been added in this release. They take an extra boolean parameter to specify whether a cancel report needs to be sent or not. If set to false, no asynchronous cancel report will be generated for the requested cancel. If set to true, behavior will be same as that of the currently existing cancel methods.

```
interface Quote : cmiV3::Quote
{
    void cancelQuoteV5(in cmiSession::TradingSessionName sessionName,
                       in cmiProduct::ProductKey productKey,
                       in boolean sendCancelReport)
    raises(
          exceptions::SystemException,
       exceptions::CommunicationException,
       exceptions::AuthorizationException,
       exceptions::DataValidationException,
       exceptions::TransactionFailedException,
       exceptions::NotAcceptedException,
       exceptions::NotFoundException
    );

    void cancelQuotesByClassV5(
                       in cmiSession::TradingSessionName sessionName,
                       in cmiProduct::ClassKey classKey,
                       in boolean sendCancelReports)
    raises(
          exceptions::SystemException,
```

```
         exceptions::CommunicationException,
         exceptions::AuthorizationException,
         exceptions::DataValidationException,
         exceptions::TransactionFailedException,
         exceptions::NotAcceptedException,
            exceptions::NotFoundException
      );

      void cancelAllQuotesV5(
                        in cmiSession::TradingSessionName sessionName,
                        in boolean sendCancelReports)
      raises(
            exceptions::SystemException,
         exceptions::CommunicationException,
         exceptions::AuthorizationException,
         exceptions::DataValidationException,
         exceptions::NotAcceptedException,
         exceptions::TransactionFailedException
         );
   };
```

IDL Interfaces

New and modified IDL is reflected in **bold** face.

```
module cmiV5
{
    interface OrderEntry : cmiV3::OrderEntry
    {
         cmiOrder::InternalizationOrderResultStruct
acceptInternalizationStrategyOrder(
            in cmiOrder::OrderEntryStruct primaryOrder,
            in cmiOrder::LegOrderEntryStructSequence
primaryOrderLegEntries,
            in cmiOrder::OrderEntryStruct matchOrder,
            in cmiOrder::LegOrderEntryStructSequence
matchOrderLegEntries,
            in cmiOrder::MatchType matchType)
```

```
                raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::DataValidationException,
                    exceptions::NotAcceptedException,
                    exceptions::TransactionFailedException
                );
    };

    interface UserTradingParameters : cmi::UserTradingParameters
    {
            cmiUtil::KeyValueStructSequence getUserRateSettings(in
string sessionName)
                raises(
                      exceptions::SystemException,
                      exceptions::CommunicationException,
                      exceptions::AuthorizationException,
                      exceptions::DataValidationException
                );
    };


    interface Quote : cmiV3::Quote
    {
        void cancelQuoteV5(
                  in cmiSession::TradingSessionName sessionName,
            in cmiProduct::ProductKey productKey,
            in boolean sendCancelReport)
                raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::DataValidationException,
                    exceptions::TransactionFailedException,
                    exceptions::NotAcceptedException,
                    exceptions::NotFoundException
```

```
                );

        void cancelQuotesByClassV5(
                  in cmiSession::TradingSessionName sessionName,
                  in cmiProduct::ClassKey classKey,
            in boolean sendCancelReports)
                raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::DataValidationException,
                    exceptions::TransactionFailedException,
                    exceptions::NotAcceptedException,
                    exceptions::NotFoundException
                );

        void cancelAllQuotesV5(
                  in cmiSession::TradingSessionName sessionName,
            in boolean sendCancelReports)
                raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::DataValidationException,
                    exceptions::NotAcceptedException,
                    exceptions::TransactionFailedException
                );
    };

    interface UserSessionManagerV5 :
cmiTradeMaintenanceService::TMSUserSessionManager
    {
        cmiV5::OrderEntry   getOrderEntryV5()
            raises(
                exceptions::SystemException,
                exceptions::CommunicationException,
                exceptions::AuthorizationException
```

```
        );

    cmiV5::UserTradingParameters getUserTradingParametersV5()
            raises(
                exceptions::SystemException,
                exceptions::CommunicationException,
                exceptions::AuthorizationException
            );

    cmiV5::Quote getQuoteV5()
        raises(
            exceptions::SystemException,
            exceptions::CommunicationException,
            exceptions::AuthorizationException
        );

};

interface UserAccessV5
{
    UserSessionManagerV5 logon(
        in cmiUser::UserLogonStruct logonStruct,
        in cmiSession::LoginSessionType sessionType,
        in cmiCallback::CMIUserSessionAdmin clientListener,
        in boolean gmdTextMessaging )
            raises(
                exceptions::SystemException,
                exceptions::CommunicationException,
                exceptions::AuthorizationException,
                exceptions::AuthenticationException,
                exceptions::DataValidationException,
                exceptions::NotFoundException
            );

 };
```

```
};

module cmiErrorCodes
{
    interface DataValidationCodes  {
        const exceptions::ErrorCode DUPLICATE_ID = 1000;
        const exceptions::ErrorCode INVALID_TIME = 1020;
        const exceptions::ErrorCode INCOMPLETE_QUOTE = 1030;
        const exceptions::ErrorCode INVALID_QUANTITY = 1040;
        const exceptions::ErrorCode INVALID_STRATEGY = 1060;
        const exceptions::ErrorCode INVALID_SPREAD = 1070;
        const exceptions::ErrorCode INVALID_USER = 1080;
        const exceptions::ErrorCode INVALID_PRODUCT = 1090;
        const exceptions::ErrorCode INVALID_PRODUCT_CLASS = 1091;
        const exceptions::ErrorCode INVALID_REPORTING_CLASS = 1092;
        const exceptions::ErrorCode INVALID_PRODUCT_DESCRIPTION = 1093;
        const exceptions::ErrorCode INVALID_OPRA_MONTH_CODE = 1094;
        const exceptions::ErrorCode INVALID_PRICE_ADJUSTMENT = 1095;
        const exceptions::ErrorCode INVALID_SESSION = 1100;
        const exceptions::ErrorCode INVALID_STATE = 1110;
        const exceptions::ErrorCode PREFERENCE_PATH_MISMATCH = 1120;
        const exceptions::ErrorCode INVALID_ORDER_ID = 1130;
        const exceptions::ErrorCode NO_WORKING_ORDER = 1135;
        const exceptions::ErrorCode LISTENER_ALREADY_REGISTERED = 1140;
        const exceptions::ErrorCode INVALID_SIDE = 1150;
        const exceptions::ErrorCode INVALID_PRICE = 1160;
        const exceptions::ErrorCode INVALID_UPDATE_ATTEMPT = 1170;
        const exceptions::ErrorCode INVALID_ORIGINATOR = 1180;
        const exceptions::ErrorCode INVALID_ACCOUNT = 1200;
        const exceptions::ErrorCode INVALID_EXECUTING_GIVEUP_FIRM =
1210;
        const exceptions::ErrorCode INVALID_CONTINGENCY_TYPE = 1220;
        const exceptions::ErrorCode INVALID_TIME_IN_FORCE = 1230;
        const exceptions::ErrorCode INVALID_POSITION_EFFECT = 1240;
        const exceptions::ErrorCode INVALID_ORIGIN_TYPE = 1250;
        const exceptions::ErrorCode INVALID_COVERAGE = 1260;
        const exceptions::ErrorCode INVALID_PRODUCT_TYPE = 1270;
```

```
        const exceptions::ErrorCode INVALID_ORDER_STATE = 1280;
        const exceptions::ErrorCode INVALID_ORDER_SOURCE = 1290;
        const exceptions::ErrorCode INVALID_BRANCH_SEQUENCE_NUMBER =
1300;
        const exceptions::ErrorCode MISSING_LISTENER = 1310;
        const exceptions::ErrorCode BUSINESS_DAY_NOT_STARTED = 1320;
        const exceptions::ErrorCode INVALID_FIELD_LENGTH = 1330;
        const exceptions::ErrorCode INVALID_STRATEGY_LEG = 1340;
        const exceptions::ErrorCode DUPLICATE_STRATEGY_LEG = 1350;
        const exceptions::ErrorCode INVALID_LEG_CONTINGENCY = 1360;
        const exceptions::ErrorCode INVALID_CANCEL_REQUEST = 1370;
        const exceptions::ErrorCode INVALID_VERSION = 1380;
        const exceptions::ErrorCode INVALID_LOGIN_MODE = 1390;
        const exceptions::ErrorCode GMD_LISTENER_ALREADY_REGISTERED =
1400;
        const exceptions::ErrorCode INVALID_TRADE_SOURCE = 1410;
        const exceptions::ErrorCode INVALID_TRADE_TYPE = 1420;
        const exceptions::ErrorCode NO_REMAINING_QUANTITY = 1430;
        const exceptions::ErrorCode INVALID_OPENING_REQUIREMENT = 1440;
        const exceptions::ErrorCode INVALID_PROCESS_NAME = 1450;
        const exceptions::ErrorCode INVALID_GROUP = 1460;
        const exceptions::ErrorCode INVALID_NAME = 1461;
        const exceptions::ErrorCode INVALID_THRESHOLD = 1462;
        const exceptions::ErrorCode INVALID_OPERATOR = 1463;
        const exceptions::ErrorCode
INVALID_TRADE_REPORT_HANDLING_INSTRUCTION = 1464;
      const exceptions::ErrorCode INVALID_OPERATION_TYPE = 1475;
      // GroupService related section
        const exceptions::ErrorCode INVALID_GROUPELEMENT = 1474;
        const exceptions::ErrorCode INVALID_GROUPELEMENT_TYPE = 1465;
        const exceptions::ErrorCode INVALID_GROUPELEMENT_RELATIONSHIP =
1466;
      const exceptions::ErrorCode GROUPELEMENT_ALREADY_EXISTS = 1467;
      const exceptions::ErrorCode GROUPRELATIONSHIP_ALREADY_EXISTS =
1468;
      const exceptions::ErrorCode INVALID_USERID_REQUESTING_CANCEL =
1469;
      const exceptions::ErrorCode INVALID_WORKSTATION_ID = 1470;
```

```
      const exceptions::ErrorCode INVALID_USERID_LIST = 1471;
      const exceptions::ErrorCode ROOT_ALREADY_EXISTS = 1472;
      const exceptions::ErrorCode INVALID_GROUP_TYPE = 1473;

        // 1500 series for additions made for linkage support
        const exceptions::ErrorCode INVALID_EXCHANGE = 1500;
        const exceptions::ErrorCode INVALID_EXTENSIONS = 1510;
        const exceptions::ErrorCode INVALID_REJECT_REQUEST = 1520;

        const exceptions::ErrorCode INVALID_ID = 1530;
        const exceptions::ErrorCode INVALID_POLICIES = 1531;
        const exceptions::ErrorCode INVALID_LIMITS = 1532;
        const exceptions::ErrorCode INVALID_TYPE = 1533;
        const exceptions::ErrorCode INVALID_COUNT_BOUNDARIES = 1534;
        const exceptions::ErrorCode INVALID_TIME_BOUNDARIES = 1535;

        // 1600 series for additions made to support internalized orders
        const exceptions::ErrorCode INVALID_MATCH_TYPE = 1600;
        const exceptions::ErrorCode INVALID_AUCTION_STATE = 1610;
        const exceptions::ErrorCode INVALID_AUCTION_ID = 1620;
        const exceptions::ErrorCode INTERNALIZATION_NOT_ALLOWED = 1630;
        const exceptions::ErrorCode INVALID_AUCTION_TYPE = 1640;
        const exceptions::ErrorCode INVALID_OPTIONAL_DATA = 1650;

        // 1700 series for additions made to support index hybrid
feature
        const exceptions::ErrorCode INVALID_CONTINGENCY_BOB_IORDER =
1700;
        const exceptions::ErrorCode INVALID_CONTINGENCY_VIX_SETTLEMENT =
1701;

        // 1710 series for  Cross Product
        const exceptions::ErrorCode UNSUPPORTED_ORIGIN_TYPE = 1711;
        const exceptions::ErrorCode UNDERLYING_LEG_NOT_LISTED_INSTOCK =
1712;
        const exceptions::ErrorCode INVALID_RATIO_FOR_CROSS_PROD = 1713;
        const exceptions::ErrorCode INVALID_LEG_STOCK_PROD_STATE = 1714;
        const exceptions::ErrorCode INVALID_CLEARING_FIRM = 1715;
```

```
        //1800 series for manual price reporting
        const exceptions::ErrorCode END_OF_SALE = 1800;
        const exceptions::ErrorCode NOT_AN_OPENING_ONLY_TRADE = 1801;
        const exceptions::ErrorCode NO_TRADE_SO_FAR = 1802;
        const exceptions::ErrorCode NOT_AN_ONLY_TRADE = 1803;
        const exceptions::ErrorCode ONLY_OPENING_TRADE_SO_FAR = 1804;
        const exceptions::ErrorCode EITHER_LAST_SALE_OR_OPENING_TRADE =
1805;
        const exceptions::ErrorCode PRICE_NOT_EQUAL_TO_LAST_SALE = 1806;
        const exceptions::ErrorCode CANCELED_VOL_NOT_CUMULATIVE_VOL =
1807;
        const exceptions::ErrorCode PRICE_NOT_EQUAL_TO_OPENING_PRICE =
1808;
        const exceptions::ErrorCode PRICE_GREATER_THAN_HIGH = 1809;
        const exceptions::ErrorCode PRICE_LESS_THAN_LOW = 1810;
        const exceptions::ErrorCode VOLUME_GREATER_THAN_CUMULATIVE_VOL =
1811;


    };

    interface AuthenticationCodes  {
        const exceptions::ErrorCode UNKNOWN_USER = 2000;
        const exceptions::ErrorCode INCORRECT_PASSWORD = 2010;
        const exceptions::ErrorCode FUNCTION_NOT_IMPLEMENTED = 2020;
        const exceptions::ErrorCode INVALID_CLIENT_LOGIN_MODE = 2030;
        const exceptions::ErrorCode USER_NOT_ENABLED = 2040;
    };

    interface CommunicationFailureCodes  {
        const exceptions::ErrorCode TRANSPORT_FAILURE = 2500;
        const exceptions::ErrorCode ROUTING_SESSION_UNAVAILABLE = 2510;
        const exceptions::ErrorCode LOST_CONNECTION = 2520;
        const exceptions::ErrorCode SERVER_NOT_AVAILABLE = 2530;
    };

    interface TransactionFailedCodes  {
```

```
        const exceptions::ErrorCode CREATE_FAILED = 3000;
        const exceptions::ErrorCode UPDATE_FAILED = 3010;
        const exceptions::ErrorCode ACTION_VETOED = 3020;
        const exceptions::ErrorCode INVALID_STATE_CHANGE = 3050;
        const exceptions::ErrorCode INCOMPLETE_STATE_CHANGE = 3060;
    };

    interface NotAcceptedCodes  {
        const exceptions::ErrorCode UNKNOWN_TYPE = 4000;
        const exceptions::ErrorCode INVALID_STATE = 4010;
        const exceptions::ErrorCode INVALID_REQUEST = 4020;
        const exceptions::ErrorCode QUOTE_RATE_EXCEEDED = 4030;
        const exceptions::ErrorCode RATE_EXCEEDED = 4040;
        const exceptions::ErrorCode SEQUENCE_SIZE_EXCEEDED = 4050;
        const exceptions::ErrorCode QUOTE_BEING_PROCESSED = 4060;
        const exceptions::ErrorCode ORDER_BEING_PROCESSED = 4070;
        const exceptions::ErrorCode EXCHANGE_CLASS_GATE_CLOSED = 4080;
        const exceptions::ErrorCode SERVER_NOT_AVAILABLE = 4090;
        const exceptions::ErrorCode ACTION_VETOED = 4100;
        const exceptions::ErrorCode QUOTE_CONTROL_ID = 4110;
        const exceptions::ErrorCode UNSUPPORTED_INTERNALIZATION = 4120;
        const exceptions::ErrorCode AUCTION_INACTIVE = 4130;
        const exceptions::ErrorCode AUCTION_ENDED = 4140;

         // If a class is being quoted by userId1 and any other userId
sharing acronym tries to
        //send quote for same class, quote will be rejected with
following error code.
        const exceptions::ErrorCode OTHER_USER_FOR_ACR_QUOTING_CLASS =
4150;

        // ErrorCodes for User Maintenance:
        //
        const exceptions::ErrorCode ONLY_USER_FOR_ACRONYM = 4200;
        const exceptions::ErrorCode USER_IS_ENABLED = 4210;
        const exceptions::ErrorCode USER_LOGGED_IN = 4220;
        const exceptions::ErrorCode USER_HAS_ORDER = 4230;
```

```
        const exceptions::ErrorCode RECENT_USER_ACTIVITY = 4240;

      //Error code for Manual Quote Reporting
      const exceptions::ErrorCode MANUAL_QUOTE_ACCEPTED = 6001;
      const exceptions::ErrorCode MANUAL_QUOTE_MARKETABLE = 6002;
      const exceptions::ErrorCode MANUAL_QUOTE_WORSE_THAN_MARKET = 6003;
      const exceptions::ErrorCode MANUAL_QUOTE_MARKETABLE_WITH_STRATEGY
= 6004;
      const exceptions::ErrorCode MANUAL_QUOTE_SYSTEM_ERROR = 6005;
      const exceptions::ErrorCode MANUAL_QUOTE_INVALID_REQUEST = 6006;
      const exceptions::ErrorCode MANUAL_QUOTE_NOT_ACCEPTED = 6007;
      const exceptions::ErrorCode MANUAL_QUOTE_OVERRIDE_NEEDED = 6008;
      const exceptions::ErrorCode
MANUAL_QUOTE_CLASS_NOT_IDX_HYBRID_ENABLED = 6009;
    };

    interface NotFoundCodes {
        const exceptions::ErrorCode RESOURCE_DOESNT_EXIST = 5000;
    };

    interface SystemCodes {
        const exceptions::ErrorCode PERSISTENCE_FAILURE = 6000;
    };

    interface AuthorizationCodes
    {
        const exceptions::ErrorCode USER_DISABLED = 7000;
        const exceptions::ErrorCode NOT_PERMITTED = 7001;
        const exceptions::ErrorCode INVALID_SESSION_ID = 7002;
        const exceptions::ErrorCode MAX_TIMEOUT_EXCEEDED = 7003;
        const exceptions::ErrorCode INVALID_PASSWORD = 7004;
        const exceptions::ErrorCode AUTHENTICATION_ERROR = 7005;
        const exceptions::ErrorCode INVALID_LOCATION = 7006;
    };

  };
```

Document Changes

API-01

- No changes

API-02

- Added a new section for AIM AON based on this release
- Included a new section" CMi V5 Quote Functionality that states:
  The existing quote cancel methods: *cancelQuote, cancelQuoteByClass* and *cancelAllQuotes* result in asynchronous quote cancel reports for all the quotes canceled. New aspects of the above quote cancel methods have been added in this release. They take an extra boolean parameter to specify whether a cancel report needs to be sent or not. If set to false, no asynchronous cancel report will be generated for the requested cancel. If set to true, behavior will be same as that of the currently existing cancel methods.
- Added a new section "Internalization Complex Order Entry" that states:
  Users can participate in a complex order auction using the new method, *acceptInternalizationStrategyOrder*. This method has been added to the *OrderEntry* interface. The new method takes two additional parameters, *primaryOrderLegEntries* and *matchOrderLegEntries*. Similar to the existing method *acceptInternalizationOrder* used for entering AIM auctions for regular orders, the new parameters specify the leg order entry structs for the primary and matching order
- Documented the new CmiV5 interfaces
- Included a new section "Retrieving the Current Rate Limits
  The *UserTradingParameters* interface was extended to include a new method, *getUserRateSettings*. This method takes a session name and returns the current values of various rate limits. Each rate limit is returned as a generic struct containing a key and a value, both of which are string types. Currently, this method returns the quote rate limit, quote call limit, order rate limit, order call limit and book depth call limit.

API-03

- Added values for the new CMi V5 interfaces and new error codes based on this release

API-04

- Added values for the new CMi V5 interfaces and new error codes based on this release

API-05

- No changes

API-06

- No changes

API-07

- No changes

API-08

- No changes

CAS-01

- No changes

CAS-02

- Updated the Order Entry section to include acceptInternalizationStrategyOrder

Simulator

- New CMiV5.0 simulator that includes a new simulator example, example16, illustrating the call to the new method, acceptInternalizationStrategyOrder.

Test Plan Changes

- No changes