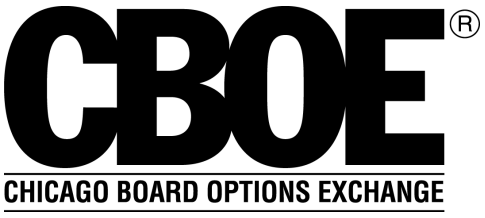

CBOE Application Programming Interface

CBOE API Version 4.2.2 - Release Notes

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

CBOE PROPRIETARY INFORMATION

01 June 2007

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 V4.2.2 HIGHLIGHTS **3**

CBSX ENHANCEMENTS 3

IDL INTERFACES **5**

DOCUMENT CHANGES **6**

API-01 6
API-02 7
API-03 7
API-04 8
API-05 8
API-06 8
API-07 8
API-08 8
CAS-01 8
CAS-02 8

SIMULATOR **8**

TEST PLAN CHANGES **8**

Overview

This document highlights upcoming changes in the new release of the CMi API, Version 4.2.2. This release supports new IDL constants and documentation changes. There are no Simulator changes. Users can continue to use the CMi V4.2 simulator. The sections below detail the changes in this release. Your feedback or questions regarding this document should be sent to api@cboe.com.

CMi V4.2.2 Highlights

CBSX Enhancements

Trade Type Indicators

CBSX enhancements include new trade type billing indicators for the W_STOCK session. A new interface, BillingTypeIndicator, will be added to the cmiConstants.idl that describes the trade types. Trade types that will be included are maker, taker, cross, flash, flash response, cross, link away, linked away response and opening.

```
module cmiConstants


interface BillingTypeIndicators
  {
    const BillingTypeIndicator MAKER                = 'A';
    const BillingTypeIndicator TAKER                = 'R';
    const BillingTypeIndicator FLASH_RESPONSE       = 'E';
    const BillingTypeIndicator FLASH                = 'F';
    const BillingTypeIndicator CROSS                = 'C';
    const BillingTypeIndicator LINKED_AWAY          = 'X';
    const BillingTypeIndicator LINKED_AWAY_RESPONSE = 'L';
    const BillingTypeIndicator OPENING              = 'O';
  };
```

Below are descriptions of the trade type indicators.

- **Maker**: refers to the person adding liquidity to the market, by basically having an order or quote resting in the book to be traded against. (i.e. they are establishing the price)
- **Taker**: refers to the person taking liquidity from the market, by basically sending an order to trade against the book. (i.e. they are coming in and taking out the best price)
- **Flash**: refers to an order that is being presented to the dealers for a short-term auction for step-up, before the order is routed to an away exchange for a fill.

- **Flash Response**: refers to the dealer responding to a flash and effectively stepping up to improve the CBSX market to the prevailing price and fulfilling the customer here.
- **Linked Away**: refers to an order that was sent to another market for execution.
- **Linked Away Response**: refers to the response from the other exchange filling the CBSX order sent to them.
- **Opening Trade**: refers to all executions that take place as part of the opening rotation process itself.
- **Cross**: refers to a trade whereby both buyer and seller are represented on a single transaction. Thus, neither is really a maker or taker per se, but rather virtually meet one another.

Fill Reports

Fill reports will be modified to include the billing type indicators in const ExtensionField BILLING_TYPE = "billingType" where “billingType” equals A, R, E, etc. This information will be reported in the FilledReportStruct.extensions.

Names Later

Traders can indicate a non real-time clearing (“Names Later”) order using const OptionalDataField NAMES_LATER = “NLTR”. If the order gets executed, the trade will not automatically clear. The trader will have to provide the contra-party information at the end of the day before the trade clears.

```
module cmiConstants

    interface OptionalDataFields
      {
        const OptionalDataField NAMES_LATER    = "NLTR";
      };
```

New Cross Orders

Two new cross order contingency types are introduced in this release, CROSS_WITHIN and TIED_CROSS_WITHIN. The CROSS_WITHIN order shall trade at or better than the NBBO and within CBOE’s market. The TIED_CROSS_WITHIN contingency allows the order to trade through the NBBO and must trade within CBOE’s market.

```
const cmiOrder::ContingencyType CROSS_WITHIN          = 22;
const cmiOrder::ContingencyType TIED_CROSS_WITHIN = 23;
```

Stock NBBO Indicator

The Stock NBBO indicator (Flash), in the W_STOCK session, will be published on the auction event channel. This is not a new auction but it is being used as an AuctionType in

order to share the event channel. If the user tries to respond with an auction, a reject message will be received.

```
const cmiOrder::AuctionType STOCK_NBBO_FLASH = 6;
```

IDL Interfaces

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

```
module cmiConstants
    interface AuctionTypes // Auction type codes
      {
        const cmiOrder::AuctionType AUCTION_INTERNALIZATION =1;
        const cmiOrder::AuctionType AUCTION_STRATEGY =2;
        const cmiOrder::AuctionType AUCTION_REGULAR_SINGLE =3;
        const cmiOrder::AuctionType AUCTION_HAL = 4;
        const cmiOrder::AuctionType AUCTION_SAL = 5;
        const cmiOrder::AuctionType AUCTION_UNSPECIFIED = 0;
        // sharing the Auction channel
        const cmiOrder::AuctionType STOCK_NBBO_FLASH = 6;
      };

    interface ContingencyTypes
      {
        const cmiOrder::ContingencyType NONE = 1; // no contingency
        const cmiOrder::ContingencyType AON = 2; // All or None
        const cmiOrder::ContingencyType FOK = 3; // Fill or Kill
        const cmiOrder::ContingencyType IOC = 4; // Immediate or Cancel
        const cmiOrder::ContingencyType OPG = 5; // Opening only
        const cmiOrder::ContingencyType MIN = 6; // Minimum
        const cmiOrder::ContingencyType NOTHELD = 7; // Not held
        const cmiOrder::ContingencyType WD = 8; // With discretion
        const cmiOrder::ContingencyType MIT = 9; // Market if touched
        const cmiOrder::ContingencyType STP = 10; // Stop order
        const cmiOrder::ContingencyType STP_LOSS = 11; // Stop loss
        const cmiOrder::ContingencyType CLOSE = 12; // On close
        const cmiOrder::ContingencyType STP_LIMIT = 13; // Stop limit
        const cmiOrder::ContingencyType AUCTION_RESPONSE = 14; // Auction response order
```

```
    const cmiOrder::ContingencyType INTERMARKET_SWEEP = 15; // Intermarket sweep
    const cmiOrder::ContingencyType RESERVE = 16; // Reserve order
    const cmiOrder::ContingencyType MIDPOINT_CROSS    = 17; // Mid Point Cross
    const cmiOrder::ContingencyType CROSS             = 18; // Cross
    const cmiOrder::ContingencyType TIED_CROSS        = 19; // Tied cross
    const cmiOrder::ContingencyType AUTOLINK_CROSS    = 20; // Auto link cross
    const cmiOrder::ContingencyType AUTOLINK_CROSS_MATCH = 21; // Auto link cross
    const cmiOrder::ContingencyType CROSS_WITHIN      = 22;
    const cmiOrder::ContingencyType TIED_CROSS_WITHIN = 23;

typedef char BillingTypeIndicator;
  interface BillingTypeIndicators
  {
    const BillingTypeIndicator MAKER              = 'A';
    const BillingTypeIndicator TAKER              = 'R';
    const BillingTypeIndicator FLASH_RESPONSE     = 'E';
    const BillingTypeIndicator FLASH              = 'F';
    const BillingTypeIndicator CROSS              = 'C';
    const BillingTypeIndicator LINKED_AWAY        = 'X';
    const BillingTypeIndicator LINKED_AWAY_RESPONSE = 'L';
    const BillingTypeIndicator OPENING            = 'O';
  };

  typedef string OptionalDataField;

  interface OptionalDataFields
  {
    const OptionalDataField NAMES_LATER       = "NLTR";
  };
```

Document Changes

API-01

- No changes

API-02

- Updated the User Input Monitor (UIM) section to state: "This feature is awaiting regulatory approval and is not currently enabled."
- Added a new section: "Determining Trade Participants."
    - The trade ID is one way to determine participants in a trade. The trade ID is unique, however, it will be the same for all orders and quotes that participated in the trade. Orders have a unique CBOE identifier (high+low) but they are not unique versus a quote ID. Currently, a given user can only have one quote per series, therefore, using the user ID is an option. For either an order or trade, you will need the transaction sequence number as there can be multiple fills generated for each.
        - To search for order fills, you will need the trade ID, OrderID.high, OrderId.low and the TransactionSequenceNumber.
        - To search for quote fills, you will need the trade ID, userID and TransactionSequenceNumber.
- Changed the order limitations for W_STOCK from 100 orders per second to 200 orders per second.
- Changed the definition of TIED_CROSS (CURRENTLY UNSUPPORTED) to read: Similar to Cross orders except that they can trade at or better than CBOE's current market and NBBO trade through is allowed.
- Add the new Cross order types CROSS_WITHIN and TIED_CROSS_WITHIN to the Cross Order section for CBSX.

    const cmiOrder::ContingencyType CROSS_WITHIN = 22; // (CURRENTLY UNSUPPORTED)

    Similar to Cross orders except that the CROSS_WITHIN order shall trade at or better than the NBBO and within CBOE's market.

    const cmiOrder::ContingencyType TIED_CROSS_WITHIN = 23; // (CURRENTLY UNSUPPORTED)

    Similar to TIED_CROSS orders except that the TIED_CROSS_WITHIN order shall trade at or better than the NBBO and within CBOE's market.
- Updated the AIM section "Consideration for Internalization" to include the A:AIR feature in the optional data field.
    - If a Firm does not wish to cancel the primary order when the auction expires, the Firm must enter A:AIR, instead of A:AIM, in the Optional Data field. This will designate the primary order to be returned to system and trade or book as a regular order.

API-03

- Added values for new constants based on this release.

API-04

- Added values for new constants 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

- No changes

Simulator

- No changes. Users can continue to use the CMi V4.2 simulator.

Test Plan Changes

- No changes