CONFIDENTIAL

CBOE Application Programming Interface

CBOE API Version 2.62 - Release Notes

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

CBOE PROPRIETARY INFORMATION

31 October 2003

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

Overview

The material presented in is document highlight the upcoming changes for the new release of the CMi API, Version 2.62. IDL, documentation and simulator changes are detailed in the sections below. Your feedback or questions regarding this document should be sent to API@cboe.com

IDL Interfaces

New and modified IDL is reflected in bold face.

CMi Intermarket Messages

```
struct FillRejectStruct
    {
        cmiUtil::CboeIdStruct tradeId;
        cmiOrder::OrderStruct order;
        long transactionSequenceNumber;
        cmiUtil::FillRejectReason rejectReason;
        string extensions;
    };
    typedef sequence <FillRejectStruct> FillRejectStructSequence;


struct PreOpeningIndicationPriceStruct
    {
        cmiIntermarketMessages::PreOpeningIndicationType preOpenType;
        cmiOrder::OriginType preOpenOriginType;
        cmiUtil::PriceStruct lowOpeningPrice;
        cmiUtil::PriceStruct highOpeningPrice;
        cmiUtil::Side side;
        long principalQuantity;
    };
    typedef  sequence <PreOpeningIndicationPriceStruct>
PreOpeningIndicationPriceStructSequence;
```

CMi Intermarket Callback

```
interface CMINBBOAgentSessionAdmin
    {
        void acceptForcedOut(
            in string reason,
            in cmiProduct::ClassKey classKey,
            in cmiSession::TradingSessionName session );

        void acceptReminder(
            in cmiIntermarketMessages::OrderReminderStruct reminder,
            in cmiProduct::ClassKey classKey,
            in cmiSession::TradingSessionName session );

        void acceptSatisfactionAlert(
```

```
        in cmiIntermarketMessages::SatisfactionAlertStruct alert,
        in cmiProduct::ClassKey classKey,
        in cmiSession::TradingSessionName session );

    void acceptIntermarketAdminMessage(
        in cmiSession::TradingSessionName session,
        in cmiUser::Exchange originatingExchange,
        in cmiProduct::ProductKeysStruct productKeys,
        in cmiIntermarketMessages::AdminStruct adminMessage);

    void acceptBroadcastIntermarketAdminMessage(
        in cmiSession::TradingSessionName session,
        in cmiUser::Exchange originatingExchange,
        in cmiIntermarketMessages::AdminStruct adminMessage);
};
```

CMi Intermarket

```
interface IntermarketManualHandling
  {
    void lockProduct(
      in cmiSession::TradingSessionName session,
      in cmiProduct::ProductKey productKey)
        raises(
          exceptions::SystemException,
          exceptions::CommunicationException,
          exceptions::AuthorizationException,
          exceptions::DataValidationException,
          exceptions::NotFoundException,
          exceptions::NotAcceptedException
        );
```

This method is for stock products only. It allows the NBBO Agent to lock the order book at a specified time before the open and up to the opening of a CBOE non-primary equity. This method can only be used before the open.

Implemented in Version 2.62

```
    void unlockProduct(
      in cmiSession::TradingSessionName session,
      in cmiProduct::ProductKey productKey)
        raises(
          exceptions::SystemException,
```

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

This method is for stock products only. It allows the NBBO Agent to unlock the order book and let queued orders enter the order book before the open. This is generally used if the primary opening is delayed. This is for CBOE non-primary equities that have not yet opened.

Implemented in Version 2.62

```
void rerouteBookedOrderToHeldOrder(
  in cmiOrder::OrderIdStruct bookedOrderId,
  in cmiSession::TradingSessionName session,
  in cmiProduct::ProductKey productKey,
  in boolean nbboProtectionFlag)
    raises(
      exceptions::SystemException,
      exceptions::CommunicationException,
      exceptions::AuthorizationException,
      exceptions::DataValidationException,
      exceptions::TransactionFailedException,
      exceptions::NotAcceptedException
  );
```

This method is for stock products only. It allows the NBBO Agent to remove an order in the order book and place it into manual handling. This method is available before the open of a CBOE non-primary equity.

Implemented in Version 2.62

```
void acceptOpeningPriceForProduct(
  in cmiUtil::PriceStruct openingPrice,
  in cmiSession::TradingSessionName session,
  in cmiProduct::ProductKey productKey)
    raises(
      exceptions::SystemException,
      exceptions::CommunicationException,
      exceptions::AuthorizationException,
```

```
        exceptions::DataValidationException,
        exceptions::TransactionFailedException,
        exceptions::NotAcceptedException
    );
```

This method is for stock products only. It allows the NBBO Agent to enter the opening price foran equity. Any order imbalance at this price will be routed to manual handling. This method is only available for use on CBOE non-primary equities.

Implemented in Version 2.62

CMi Util

```
module cmiUtil
{

  typedef string VersionLabel;
  typedef short PriceType;
  typedef sequence <PriceType> PriceTypeSequence;
  typedef char  EntryType;
  typedef char  Side;
  typedef char  Source;
  typedef short UpdateStatusReason;
  typedef short ActivityReason;
  typedef short QueryDirection;
  typedef sequence <string> StringSequence;
  typedef sequence <long> LongSequence;
  typedef double PricingModelParameter;
  typedef short ReportType;
  typedef short OrderFlowDirection;
  typedef short QueueAction;
  typedef short LinkageMechanism;
  typedef short SatisfactionOrderDisposition;
  typedef short SatisfactionOrderRejectReason;
  typedef short FillRejectReason;

  struct DateStruct
  {
    octet month;
    octet day;
    short year;
  };
  typedef sequence< DateStruct > DateStructSequence;

  struct TimeStruct
  {
    octet hour;
    octet minute;
    octet second;
    octet fraction;
  };
  typedef sequence< TimeStruct > TimeStructSequence;
```

```
    struct DateTimeStruct
    {
        cmiUtil::DateStruct date;
        cmiUtil::TimeStruct time;
    };
    typedef sequence< DateTimeStruct > DateTimeStructSequence;

    struct PriceStruct
    {
        cmiUtil::PriceType type;
        long whole;
        long fraction;
    };
    typedef sequence< PriceStruct > PriceStructSequence;


    struct CallbackInformationStruct
    {
        string subscriptionInterface;
        string subscriptionOperation;
        string subscriptionValue;
        string ior;
    };

    struct CboeIdStruct
    {
        long highCboeId;
        long lowCboeId;
    };

};
```

CMi Constants

```
interface ActivityTypes
    {

        // New Activity Types for Linkage
        const cmiTraderActivity::ActivityType NEW_ORDER_REJECT = 301;
        const cmiTraderActivity::ActivityType FILL_REJECT = 302;
        const cmiTraderActivity::ActivityType CANCEL_ORDER_REQUEST = 303;
        const cmiTraderActivity::ActivityType CANCEL_ORDER_REQUEST_REJECT = 304;
        const cmiTraderActivity::ActivityType CANCEL_REPORT_REJECT = 305;
        const cmiTraderActivity::ActivityType NEW_ORDER_REJECT_REJECTED = 306;
        const cmiTraderActivity::ActivityType FILL_REJECT_REJECTED = 307;
        const cmiTraderActivity::ActivityType
CANCEL_ORDER_REQUEST_REJECT_REJECTED = 308;
        const cmiTraderActivity::ActivityType CANCEL_REPORT_REJECT_REJECTED = 309;
        const cmiTraderActivity::ActivityType ROUTE_TO_AWAY_EXCHANGE = 310;
    };
```

```
interface ExtensionFields
   {

      // Used for routing an order to a BART terminal

      const ExtensionField BARTID = "BARTID";
      // Firm information for stock leg of a buy-write

      const ExtensionField STOCK_FIRM = "STOCK_FIRM";
      const ExtensionField STOCK_FIRM_NAME = "STOCK_FIRM_NAME";

      // The following are used for linkage

      const ExtensionField CBOE_EXEC_ID = "cboeExecId";
      const ExtensionField ORIGINAL_QUANTITY = "originalQuantity";
      const ExtensionField SIDE = "side";
      const ExtensionField EXEC_BROKER = "execBroker";
      const ExtensionField ORS_ID = "orsId";
      const ExtensionField SATISFACTION_ALERT_ID = "satAlertId";
      const ExtensionField ASSOCIATED_ORDER_ID = "assocOrderId";
      const ExtensionField LINKAGE_MECHANISM = "LinkageMechanism";
      const ExtensionField EXPIRATION_TIME = "ExpirationTime";

      const ExtensionField AWAY_CANCEL_REPORT_EXEC_ID="awayCancelReportExecId";
      const ExtensionField AWAY_EXCHANGE_USER_ACRONYM="1";
      const ExtensionField USER_ASSIGNED_CANCEL_ID="11";
      const ExtensionField AWAY_EXCHANGE_EXEC_ID="17";
      const ExtensionField HANDLING_INSTRUCTION="21";
      const ExtensionField AWAY_EXCHANGE_ORDER_ID = "37";
      const ExtensionField TEXT = "58";
      const ExtensionField AWAY_EXCHANGE_TRANSACT_TIME = "60";
      const ExtensionField EXCHANGE_DESTINATION = "100";
      const ExtensionField AUTO_EXECUTION_SIZE = "5201";
      const ExtensionField TRADE_THRU_TIME = "5202";
      const ExtensionField TRADE_THRU_SIZE = "5203";
      const ExtensionField TRADE_THRU_PRICE = "5204";
      const ExtensionField ADJUSTED_PRICE_INDICATOR = "5205";
      const ExtensionField SATISFACTION_ORDER_DISPOSITION = "5206";
      const ExtensionField EXECUTION_RECEIPT_TIME = "5207";
      const ExtensionField ORIGINAL_ORDER_TIME = "5208";
      const ExtensionField OLA_REJECT_REASON = "5209";
      const ExtensionField ORDER_CAPACITY = "6528";
      const ExtensionField ORDER_RESTRICTIONS = "6529";
   };

interface PriceTypes

   {

      const cmiUtil::PriceType NO_PRICE = 1;
      const cmiUtil::PriceType LIMIT = 2;
```

```
      const cmiUtil::PriceType VALUED = 2;
      const cmiUtil::PriceType MARKET = 3;
      const cmiUtil::PriceType CABINET = 4;
   };

interface ExchangeStrings
   {
   const cmiUser::Exchange AMEX    = "AMEX";   //American Stock Exchange
   const cmiUser::Exchange BSE     = "BSE";    //Boston Stock Exchange
   const cmiUser::Exchange CBOE    = "CBOE";   //Chicago Board Options Exchange
   const cmiUser::Exchange CBOT    = "CBOT";   //Chicago Board of Trade
   const cmiUser::Exchange CHX     = "CHX";    //Chicago Stock Exchange
   const cmiUser::Exchange CME     = "CME";    //Chicago Mercantile Exchange
   const cmiUser::Exchange CSE     = "CSE";    //Cincinnati Stock Exchange
   const cmiUser::Exchange ISE     = "ISE";    //International Stock Exchange
   const cmiUser::Exchange LIFFE   = "LIFFE";  //International Financial Futures and Options
Exchange
   const cmiUser::Exchange NASD    = "NASD";   //National Association of Securities Dealers
   const cmiUser::Exchange NYME    = "NYME";   //New York Mercantile Exchange
   const cmiUser::Exchange NYSE    = "NYSE";   //New York Stock Exchange
   const cmiUser::Exchange ONE     = "ONE";    //OneChicago Exchange
   const cmiUser::Exchange PHLX    = "PHLX";   //Philadelphia Stock Exchange
   const cmiUser::Exchange PSE     = "PSE";    //Pacific Stock Exchange
   const cmiUser::Exchange NQLX    = "NQLX";      //Nasdaq Liffe Markets
  const cmiUser::Exchange BOX    = "BOX";    // Boston Options Exchange
      const cmiUser::Exchange CFE    = "CFE";      //CBOE Futures Exchange
   };


interface AlertResolutions
   {
        const cmiIntermarketMessages::AlertResolution NOT_RESOLVED ="NR";
        const cmiIntermarketMessages::AlertResolution ADJUSTED="A";
        const cmiIntermarketMessages::AlertResolution PARTIAL_PRICE_ADJUSTMENT="AP";
        const cmiIntermarketMessages::AlertResolution PARTIAL_QUANTITY_ADJ="AQ";
        const cmiIntermarketMessages::AlertResolution CONTRA_UNAVAILABLE="CU";
        const cmiIntermarketMessages::AlertResolution DELAYED_REPORT="DR";
        const cmiIntermarketMessages::AlertResolution ERRONEOUS_REPORT="ER";
        const cmiIntermarketMessages::AlertResolution FIRM_DISCRETION="FD";
        const cmiIntermarketMessages::AlertResolution
  EXECUTED_UNDER_FIRM_INSTRUCTIONS="FI";
        const cmiIntermarketMessages::AlertResolution FAST_MARKET_AWAY="FM";
        const cmiIntermarketMessages::AlertResolution FLASH_QUOTE="FQ";
        const cmiIntermarketMessages::AlertResolution
  AWAY_MARKET_UNAVAIL_TO_TRADE="IN";
        const cmiIntermarketMessages::AlertResolution NBBO_LOCKED_W_CBOE="LB";
        const cmiIntermarketMessages::AlertResolution NBBO_FADE="NF";
        const cmiIntermarketMessages::AlertResolution
  AWAY_MARKET_REFUSE_TO_TRADE_OR_FADE="NU";
        const cmiIntermarketMessages::AlertResolution OTHER="O";
        const cmiIntermarketMessages::AlertResolution POST_TRADE_QUOTE="PQ";
        const cmiIntermarketMessages::AlertResolution SHUT_OFF_ERROR="SE";
        const cmiIntermarketMessages::AlertResolution SINGLE_LISTED_OPTION="SL";
```

```
        const cmiIntermarketMessages::AlertResolution CBOE_SYSTEM_PROBLEMS="SP";
        const cmiIntermarketMessages::AlertResolution TRADE_BUSTED="TB";
        const cmiIntermarketMessages::AlertResolution NOT_ADJUSTED="UA";
        const cmiIntermarketMessages::AlertResolution
    BOOK_TAKEN_OUT_AFTER_NOTIFICATION="BA";
        const cmiIntermarketMessages::AlertResolution
    TRADE_ENTERED_ON_REFRESHED_QUOTE="TO";
     };

interface SatisfactionOrderRejectReasons
     {
        const cmiUtil::SatisfactionOrderRejectReason LATE_PRINT =
    ActivityReasons::LATE_PRINT;
        const cmiUtil::SatisfactionOrderRejectReason COMM_DELAYS =
    ActivityReasons::COMM_DELAYS;
        const cmiUtil::SatisfactionOrderRejectReason CROWD_TRADE =
    ActivityReasons::CROWD_TRADE;
        const cmiUtil::SatisfactionOrderRejectReason PROCESSING_PROBLEMS =
    ActivityReasons::PROCESSING_PROBLEMS;
        const cmiUtil::SatisfactionOrderRejectReason INVALID_PRODUCT_TYPE =
    ActivityReasons::INVALID_PRODUCT_TYPE;
        const cmiUtil::SatisfactionOrderRejectReason TRADE_REJECTED =
    ActivityReasons::TRADE_REJECTED;
        const cmiUtil::SatisfactionOrderRejectReason TRADE_BUSTED =
    ActivityReasons::TRADE_BUSTED;
        const cmiUtil::SatisfactionOrderRejectReason ORIGINAL_ORDER_REJECTED =
    ActivityReasons::ORIGINAL_ORDER_REJECTED;
        const cmiUtil::SatisfactionOrderRejectReason NON_BLOCK_TRADE =
    ActivityReasons::NON_BLOCK_TRADE;
     };

  interface FillRejectReasons
     {
        const cmiUtil::FillRejectReason INVALID_PRODUCT =
    ActivityReasons::INVALID_PRODUCT;
        const cmiUtil::FillRejectReason INVALID_SIDE =  ActivityReasons::INVALID_SIDE;
        const cmiUtil::FillRejectReason INVALID_QUANTITY =
    ActivityReasons::INVALID_QUANTITY;
        const cmiUtil::FillRejectReason NO_MATCHING_ORDER =
    ActivityReasons::NO_MATCHING_ORDER;
        const cmiUtil::FillRejectReason INVALID_PRICE =  ActivityReasons::INVALID_PRICE;
        const cmiUtil::FillRejectReason STALE_EXECUTION =
    ActivityReasons::STALE_EXECUTION;
        const cmiUtil::FillRejectReason OTHER=  ActivityReasons::OTHER;
     };
```

CAS Simulator Changes

- New Java examples and functionality for Stock

Document Changes

API-01

- No changes

API-02

- New examples for Stock

API-03

- Added new constants, structs and methods based on this IDL release

API-04

- Added definitions for the new constants, structs and methods reflected in this IDL release

API-05

- JDK Orb changes

API-06

- No changes

API-07

- No changes

CAS-01

- No changes

CAS-02

- New functionality for Stock