CBOE Application Programming Interface

CBOE API Version 4.0 - Release Notes

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

CBOE PROPRIETARY INFORMATION

25 May 2006

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

This document highlights changes for the new release of the CMi API, Version 4.0. This release supports new features for CBOE's Market Data Express (MDX) data feed. 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.

A new API document has been created for users who are specifically interested in MDX. Refer to CBOE's API Volume 8 (API-08): "CMi Programmer's Guide to the Market Data Express (MDX) Data Feed" to review the MDX application design.

CMi V4.0 Highlights

The strategy of MDX is to expand the distribution of the market data currently available. The CBOE is adding a new interface, cmiV4, to support the MDX data feed. This interface is designed to generically support the reporting of quotes, trades, and other information for options, stocks and futures.

Market Data

The new ticker, recap and last sale callbacks now feature a sequence number in the unlikely event that a message arrives out of sequence. The existing recap callback is broken up into two separate callbacks: recap and last sale. New subscription methods are provided in the new MarketQuery interface.

CAS Oneway Callbacks

CAS oneway callbacks are high-performance calls that function differently than a standard CORBA call. Normally, a CORBA function call will wait for the full round trip to complete before continuing. A oneway CORBA call will not wait for any acknowledgement from the server before continuing.

The exact effect this has on your application will depend on your ORB's POA threading policy. If your POA is single-threaded then things will work as they were before with calls queuing up on a single thread. If your POA is multi-threaded then there is no guarantee what order the calls will be received. A sequence number is provided in the method signatures as an aid to determine whether a message was received out of order. Regardless of the threading policy it is essential to keep track of the current sequence number and compare incoming messages.

Session Management

The new CMi interfaces will be made available as extensions to the existing CMi on the existing CAS. A CMi user can gain access to the enhanced interfaces by getting a reference to the UserSessionManagerV4. This reference can be obtained through logon using the UserAccessV4 interface. The new logon method takes exactly the same parameters as the old CMi's logon method and returns a newly enhanced SessionManagerStructV4. The UserAccessV4 object will be made available as an alternate IOR link on the HTTP port that the CAS is publishing on.

IDL Interfaces

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

cmiV4.idl

```
module cmiV4
{
   interface MarketQuery    {
       void subscribeCurrentMarket(
         in cmiProduct::ClassKey classKey,
         in cmiCallbackV4::CMICurrentMarketConsumer clientListener,
         in cmiUtil::QueueAction actionOnQueue)
            raises(
               exceptions::SystemException,
               exceptions::CommunicationException,
               exceptions::AuthorizationException,
               exceptions::DataValidationException
            );

         void unsubscribeCurrentMarket(
         in cmiProduct::ClassKey classKey,
         in cmiCallbackV4::CMICurrentMarketConsumer clientListener)
            raises(
               exceptions::SystemException,
               exceptions::CommunicationException,
               exceptions::AuthorizationException,
               exceptions::DataValidationException
            );

        void subscribeTicker(
         in cmiProduct::ClassKey classKey,
         in cmiCallbackV4::CMITickerConsumer clientListener,
         in cmiUtil::QueueAction actionOnQueue)
            raises(
               exceptions::SystemException,
               exceptions::CommunicationException,
               exceptions::AuthorizationException,
               exceptions::DataValidationException
            );

        void unsubscribeTicker(
         in cmiProduct::ClassKey classKey,
         in cmiCallbackV4::CMITickerConsumer clientListener)
            raises(
               exceptions::SystemException,
               exceptions::CommunicationException,
               exceptions::AuthorizationException,
               exceptions::DataValidationException
            );
```

```
        void subscribeRecap(
         in cmiProduct::ClassKey classKey,
         in cmiCallbackV4::CMIRecapConsumer clientListener,
         in cmiUtil::QueueAction actionOnQueue)
            raises(
               exceptions::SystemException,
               exceptions::CommunicationException,
               exceptions::AuthorizationException,
               exceptions::DataValidationException
            );

        void unsubscribeRecap(
         in cmiProduct::ClassKey classKey,
         in cmiCallbackV4::CMIRecapConsumer clientListener)
            raises(
               exceptions::SystemException,
               exceptions::CommunicationException,
               exceptions::AuthorizationException,
               exceptions::DataValidationException
            );
    };

    interface UserSessionManagerV4 : cmiV3::UserSessionManagerV3  {
       cmiV4::MarketQuery   getMarketQueryV4()
          raises(
             exceptions::SystemException,
             exceptions::CommunicationException,
             exceptions::AuthorizationException
          );
    };

    interface UserAccessV4
    {
       UserSessionManagerV4 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
             );

     };
};
```

The cmiV4 module contains the interfaces to Version 4 application server services. It contains the interfaces for requests between client and the CAS. References to the application server services objects are obtained via the SessionManager. A reference to the SessionManager is obtained via the user access service.

cmiCallbackV4.idl

```
module cmiCallbackV4
{
 interface CMICurrentMarketConsumer {
      oneway void acceptCurrentMarket(
         in cmiMarketData::CurrentMarketStructV4Sequence bestMarkets,
         in cmiMarketData::CurrentMarketStructV4Sequence bestPublicMarkets,
         in long messageSequence,
         in long queueDepth,
         in cmiUtil::QueueAction queueAction);
  };

 interface CMITickerConsumer {
      oneway void acceptTicker(
         in cmiMarketData::TickerStructV4Sequence ticker,
         in long messageSequence,
         in long queueDepth,
         in cmiUtil::QueueAction queueAction);
  };

 interface CMIRecapConsumer {
      oneway void acceptRecap(
         in cmiMarketData::RecapStructV4Sequence recap,
         in long messageSequence,
         in long queueDepth,
         in cmiUtil::QueueAction queueAction);

      oneway void acceptLastSale(
         in cmiMarketData::LastSaleStructV4Sequence lastSale,
         in long messageSequence,
         in long queueDepth,
         in cmiUtil::QueueAction queueAction);
  };
};
```

All CBOE Market Interface call back interfaces for Version 4.0 are contained within this module. The new oneway callbacks are higher performance than previous versions. In addition, the cmiCallbackV4 provides a sequence number to check for messages received in proper order.

cmiMarketData.idl

```
module cmiMarketData
{
   typedef short ExpectedOpeningPriceType;
   typedef short MarketDataHistoryEntryType;
   typedef short MarketChangeReason;
   typedef short VolumeType;
   typedef short CurrentMarketViewType;
   typedef char BookDepthUpdateType;
   typedef char TickDirectionType;
   typedef char MarketIndicator;
```

```
typedef short OrderBookPriceViewType;
typedef octet CurrentMarketType;
typedef long IntegerTime;
typedef long IntegerPrice;
typedef octet MultiplePartiesIndicator;


struct MarketVolumeStructV4 {
   cmiMarketData::VolumeType volumeType;
   long quantity;
   cmiMarketData::MultiplePartiesIndicator multipleParties;
};
typedef sequence <MarketVolumeStructV4> MarketVolumeStructV4Sequence;

struct CurrentMarketStructV4
{
   cmiProduct::ClassKey classKey;
   cmiProduct::ProductKey productKey;
   cmiProduct::ProductType productType;
   string exchange;
   cmiMarketData::IntegerTime sentTime;

   cmiMarketData::CurrentMarketType currentMarketType;
   cmiMarketData::IntegerPrice bidPrice;
                cmiMarketData::TickDirectionType bidTickDirection;
   cmiMarketData::MarketVolumeStructV4Sequence bidSizeSequence;
   cmiMarketData::IntegerPrice askPrice;
   cmiMarketData::MarketVolumeStructV4Sequence askSizeSequence;
   cmiMarketData::MarketIndicator marketIndicator;
   cmiSession::ProductState productState;
   octet priceScale;
};
typedef sequence <CurrentMarketStructV4> CurrentMarketStructV4Sequence;

struct TickerStructV4
{
   cmiProduct::ClassKey classKey;
   cmiProduct::ProductKey productKey;
   cmiProduct::ProductType productType;
   string exchange;
   cmiMarketData::IntegerTime sentTime;

   octet priceScale;
   cmiMarketData::IntegerTime tradeTime;
   cmiMarketData::IntegerPrice tradePrice;
   long tradeVolume;
   string salePrefix;
   string salePostfix;
 };
 typedef sequence <TickerStructV4> TickerStructV4Sequence;

struct RecapStructV4
{
   cmiProduct::ClassKey classKey;
   cmiProduct::ProductKey productKey;
```

```
      cmiProduct::ProductType productType;
      string exchange;
      cmiMarketData::IntegerTime sentTime;

      octet priceScale;
      cmiMarketData::IntegerPrice lowPrice;
      cmiMarketData::IntegerPrice highPrice;
      cmiMarketData::IntegerPrice openPrice;
      cmiMarketData::IntegerPrice previousClosePrice;
      string statusCodes;
   };
   typedef sequence <RecapStructV4> RecapStructV4Sequence;

   struct LastSaleStructV4
   {
      cmiProduct::ClassKey classKey;
      cmiProduct::ProductKey productKey;
      cmiProduct::ProductType productType;
      string exchange;
      cmiMarketData::IntegerTime sentTime;

      octet priceScale;
      cmiMarketData::IntegerTime lastSaleTime; // millis since midnight
      cmiMarketData::IntegerPrice lastSalePrice;
      long lastSaleVolume;
      long totalVolume;
      cmiMarketData::TickDirectionType tickDirection;
      cmiMarketData::IntegerPrice netPriceChange;
   };
   typedef sequence <LastSaleStructV4> LastSaleStructV4Sequence;
};
```

cmiConstants.idl

```
module cmiConstants
{
   interface LoginSessionModes
   {
      const cmiUser::LoginSessionMode STAND_ALONE_TEST = '1';
      const cmiUser::LoginSessionMode NETWORK_TEST = '2';
      const cmiUser::LoginSessionMode PRODUCTION = '3';
   };
   interface CurrentMarketTypes
   {
      const cmiMarketData::CurrentMarketType BEST_MARKETS = 1;
      const cmiMarketData::CurrentMarketType BEST_PUBLIC_MARKETS = 2;
   };
```

```
interface ProductStates
 {
    const cmiSession::ProductState UNKNOWN = 0;
    const cmiSession::ProductState CLOSED = 1;
    const cmiSession::ProductState PRE_OPEN = 2;
    const cmiSession::ProductState OPENING_ROTATION = 3;
    const cmiSession::ProductState OPEN = 4;
    const cmiSession::ProductState HALTED = 5;
    const cmiSession::ProductState FAST_MARKET = 6;
    const cmiSession::ProductState NO_SESSION = 7;
    const cmiSession::ProductState ON_HOLD = 8;
    const cmiSession::ProductState ENDING_HOLD = 9;
    const cmiSession::ProductState SUSPENDED = 10;
 };

interface TickDirectionTypes
 {
    const cmiMarketData::TickDirectionType PLUS_TICK       = '+';
    const cmiMarketData::TickDirectionType MINUS_TICK      = '-';
    const cmiMarketData::TickDirectionType ZERO_MINUS_TICK = '_';
    const cmiMarketData::TickDirectionType ZERO_PLUS_TICK  = '*';
    const cmiMarketData::TickDirectionType UNKNOWN_TICK    = ' ';
 };

 interface PriceConstants
 {
   const cmiMarketData::IntegerPrice NO_PRICE = -2147483648;
 };

 interface MarketIndicators
 {
    const cmiMarketData::MarketIndicator REGULAR_QUOTE          = 1;
    const cmiMarketData::MarketIndicator AUTO_EXECUTION         = 2;
    const cmiMarketData::MarketIndicator BID_IS_BOOK            = 3;
    const cmiMarketData::MarketIndicator ASK_IS_BOOK            = 4;
    const cmiMarketData::MarketIndicator BID_ASK_IS_BOOK        = 5;
    const cmiMarketData::MarketIndicator INACTIVE               = 6;
    const cmiMarketData::MarketIndicator ROTATION               = 7;
    const cmiMarketData::MarketIndicator FAST_MARKET            = 8;
    const cmiMarketData::MarketIndicator TRADING_HALT           = 9;
    const cmiMarketData::MarketIndicator DISQUALIFIED           = 10;
    const cmiMarketData::MarketIndicator UNKNOWN                = 11;
    const cmiMarketData::MarketIndicator DEPTH_ON_OFFER         = 12;
    const cmiMarketData::MarketIndicator DEPTH_ON_BID           = 13;
    const cmiMarketData::MarketIndicator CLOSING                = 14;
    const cmiMarketData::MarketIndicator NEWS_DISSEMINATION     = 15;
    const cmiMarketData::MarketIndicator ORDER_INFLUX           = 16;
    const cmiMarketData::MarketIndicator PRE_OPENING_INDICATION = 17;
    const cmiMarketData::MarketIndicator DEPTH_BID_OFFER        = 18;
    const cmiMarketData::MarketIndicator ORDER_IMBALANCE        = 19;
    const cmiMarketData::MarketIndicator HALT_REL_SEC_NEWS_DISS = 20;
    const cmiMarketData::MarketIndicator HALT_REL_SEC_NEWS_PEND = 21;
```

```
    const cmiMarketData::MarketIndicator CLOSED_MARKET_MAKER      = 22;
    const cmiMarketData::MarketIndicator ADDITIONAL_INFORMATION   = 23;
    const cmiMarketData::MarketIndicator NON_FIRM_QUOTE           = 24;
    const cmiMarketData::MarketIndicator OPENING                  = 25;
    const cmiMarketData::MarketIndicator NEWS_PENDING             = 26;
    const cmiMarketData::MarketIndicator ADDITIONAL_INFO_REL_SEC  = 27;
    const cmiMarketData::MarketIndicator HALT_FOR_RELATED_SEC     = 28;
    const cmiMarketData::MarketIndicator RESUME                   = 29;
    const cmiMarketData::MarketIndicator HALT_IN_VIEW_COMMON      = 30;
    const cmiMarketData::MarketIndicator EQUIPMENT_CHANGOVER      = 31;
    const cmiMarketData::MarketIndicator NO_OPEN_RESUME           = 32;

};

  interface MultiplePartiesIndicators
  {
    const cmiMarketData::MultiplePartiesIndicator YES     = 1;
    const cmiMarketData::MultiplePartiesIndicator NO      = 2;
    const cmiMarketData::MultiplePartiesIndicator UNKNOWN = 3;
  };

};
```

Document Changes

API-01

- Referenced the new MDX document, API-08.

API-02

- Changed the source code to read:

  ${INSTALL_DIR}\Examples\src\javapoa\com\cboe\examples\example1\example1.java

  Where:

  ${INSTALL_DIR} is the install directory you selected during the installation of the CBOE Software Development Kit. The default location is *C:\CBOE\CMi4.0.*

  The C++ version of the example program can be found in:

  ${INSTALL_DIR}\Examples\src\cpppoa

  Note that all C++ code is kept in the same directory.

- Removed Linux as a supported development platform.
- Revised list of CORBA products to read:
  - Java 1.4 JDK from Sun Microsystems (only for testing, not production)
  - omniORB from AT&T Laboratories Cambridge

  - Tao Orb from Object Computing or Washington University
  - JacORB 2.2.3 from www.jacorb.org.
- Added tables for cmiV4 interfaces and cmiV4 callback interfaces.
- Inserted a new section "Oneway CORBA Calls for V4.0. The section reads:

The cmiCallbackV4 callbacks are used for CBOE's Market Data Express (MDX) data feed. The callbacks are implemented as oneway CORBA calls. These are high-performance calls that function differently than a standard CORBA call. Normally a CORBA function call will wait for the full round trip to complete before continuing. A oneway CORBA call will not wait for any acknowledgement from the server before continuing.

The exact effect this has on your application will depend on your ORB's POA threading policy. If your POA is single-threaded then things will work as they were before with calls queuing up on a single thread. If your POA is multi-threaded then there is no guarantee what order the calls will be received. A sequence number is provided in the method signatures as an aid to determine whether a message was received out of order. Regardless of the threading policy it is essential to keep track of the current sequence number and compare incoming messages to it.

Refer to document API-08, the "CMi Programmer's Guide to the Market Data Express (MDX) Data Feed," for complete details on the use of oneway CORBA calls for MDX.

- Added a new sub-section "Market Data V4 Functionality" under the Market Data section. The section reads:

  The new ticker, recap and last sale callbacks now feature a sequence number in the unlikely event that a message arrives out of sequence. The existing recap callback is broken up into two separate callbacks: recap and last sale. New subscription methods are provided in the new MarketQuery interface.

  Refer to document API-08, the "CMi Programmer's Guide to the Market Data Express (MDX) Data Feed," for complete details on the use of market data for MDX.

- Changed the section "Session Management V3 Functionality" to Session Management V3 and V4 Functionality. The section reads as follows.

  The CMi V3 and V4 interfaces will be made available as extensions to the existing CMi on the existing CAS. A CMi user can gain access to the enhanced interfaces by getting a reference to the UserSessionManagerV3 or UserSessionManagerV4. This reference can be obtained through logon using the UserAccessV3 and UserAccessV4 interfaces respectively. The new logon method takes exactly the same parameters as the old CMi's logon method and returns a newly enhanced SessionManagerStructV3 or SessionManagerStuctV4. The UserAccessV3 or UserAccesV4 objects will be made available as an alternate IOR link on the HTTP port that the CAS is publishing on.

  Refer to document API-08, the "CMi Programmer's Guide to the Market Data Express (MDX) Data Feed," for complete details on using the UserSessionManagerV4 for MDX.

- Added Example 15, Current Market V4.

- Changed the Quote Limit section to read: "Quote and Order Limits" and updated it to the following:

  The limitations for quoting are:

  CBOE does not consider quote or mass quote cancels when calculating quote thresholds.

  Hybrid (in session W_MAIN)

  - 40 Quotes (products) per Mass Quote message **(SEQUENCE_SIZE_EXCEEDED)**
  - 100 Mass Quote or Quote message calls per user per 3 second period **(RATE_EXCEEDED)**
  - 3000 total quotes (products) per user per 3 second period **(QUOTE_RATE_EXCEEDED)**

  CFE_MAIN (CBOE Futures Exchange) and **ONE_MAIN** (OneChicago)

  - 4 Quotes (products) per Mass Quote message
  - 50 Mass Quote or Quote message calls per user per one (1) second period
  - 1000 total quotes (products) per user per five (5) second period

  The limitations for orders are:

  Hybrid (in session W_MAIN)

  - 30 orders per one (1) second period

  CFE_MAIN (CBOE Futures Exchange) and **ONE_MAIN** (OneChicago)

  - 30 orders per one (1) second period

API-03

- Referenced the new MDX document, API-08.
- Added new messages and data types based on the IDL in is release.

API-04

- Referenced the new MDX document, API-08.
- Added new attributes and operations for the V4 interfaces based on this release.

API-05

- Removed support of Sun's JDK Orb and Visibroker
- Added support for Tao 1.4 and JacOrb 2.2.3

API-06

- Referenced the new MDX document, API-08.

API-07

- Referenced the new MDX document, API-08.

API-08

- New MDX document, API-08.

CAS-01

- Referenced the new MDX document, API-08.

CAS-02

- Updated the document to include the removal of callbacks from test programs
- Captured screen shots to include V4 interfaces

Simulator

- Current Market, Ticker, Recap, Last Sale example
- MDX Data generator on/off switch

Test Plan Changes

- No changes