CHICAGO BOARD OPTIONS EXCHANGE

# CBOE Application Programming Interface

## CBOE API Version 3.0 - Release Notes

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

***CBOE PROPRIETARY INFORMATION***

18 June 2004

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 3.0. This release supports updates for Stock trading on CBOEdirect as well as functional upgrades for the Hybrid Trading environment. IDL, documentation and simulator changes for the CMi V3.0 are detailed in the sections below. Your feedback or questions regarding this document should be sent to api@cboe.com.

Firms wishing to connect to the current CMi API production system should use the Version 2.52 documents and IDL. The documentation and IDL is available for download on the API web site at http://systems.cboe.com/webAPI/.

Below are descriptions of the current CMi API releases that are available for download on the API website at http://systems.cboe.com/webAPI/.

- V2.5 – Current production version, including simulator.
- V2.52 – Current production upgrade of constants and error codes, no simulator
- V2.62 – Stock simulator
- V3.0 – Hybrid and Stock updates with simulator – ***this release***

# CMi V3.0 Highlights

## Market Data

The existing API provides the top of the book information on the callbacks. This release provides a public market parameter on the new callbacks. This value will inform users of the Customer and Professional size if, and only if, there is Customer and/or Professional interest at the Top of the Book. To access this information, new subscription methods are provided in the new MarketQuery interface.

CMi CallbackV3

```
module cmiCallbackV3
{
   interface CMICurrentMarketConsumer {
      void acceptCurrentMarket(
         in cmiMarketData::CurrentMarketStructSequence bestMarkets,
              in cmiMarketData::CurrentMarketStructSequence bestPublicMarket,
         in long queueDepth,
         in cmiUtil::QueueAction queueAction);
   };
};
```

CMi V3

module cmiV3

```
interface MarketQuery : cmiV2::MarketQuery
  {
    void subscribeCurrentMarketForClassV3(
      in cmiSession::TradingSessionName sessionName,
      in cmiProduct::ClassKey classKey,
      in cmiCallbackV3::CMICurrentMarketConsumer clientListener,
      in cmiUtil::QueueAction actionOnQueue)
        raises(
          exceptions::SystemException,
          exceptions::CommunicationException,
          exceptions::AuthorizationException,
          exceptions::DataValidationException
        );

    void unsubscribeCurrentMarketForClassV3(
      in cmiSession::TradingSessionName sessionName,
      in cmiProduct::ClassKey classKey,
      in cmiCallbackV3::CMICurrentMarketConsumer clientListener)
        raises(
          exceptions::SystemException,
          exceptions::CommunicationException,
          exceptions::AuthorizationException,
          exceptions::DataValidationException
        );

    void subscribeCurrentMarketForProductV3(
      in cmiSession::TradingSessionName sessionName,
      in cmiProduct::ProductKey productKey,
      in cmiCallbackV3::CMICurrentMarketConsumer clientListener,
      in cmiUtil::QueueAction actionOnQueue)
        raises(
          exceptions::SystemException,
          exceptions::CommunicationException,
```

```
            exceptions::AuthorizationException,
            exceptions::DataValidationException
        );

    void unsubscribeCurrentMarketForProductV3(
        in cmiSession::TradingSessionName sessionName,
        in cmiProduct::ProductKey productKey,
        in cmiCallbackV3::CMICurrentMarketConsumer clientListener)
          raises(
            exceptions::SystemException,
            exceptions::CommunicationException,
            exceptions::AuthorizationException,
            exceptions::DataValidationException
        );
};
```

## Quote

There are several new quote functionalities in this release.

A new field has been added to the quote message to indicate that a quote cancellation should be overridden. This is to solve the cases when a market maker's quote system is in the process of sending in new quotes at the same time that CBOEdirect is removing the old quotes. The messages may cross in flight and the market maker will have replaced his old quotes. One particular example is QRM. The QRM is intended to protect a market maker from being hit multiple times across a large number of series. The market maker is streaming their quotes into the market without knowing that the QRM has triggered. This field will provide the user with the ability to decide to protect against new quotes being processed before the QRM notice has been acted upon.

1. When a market maker begins sending quotes in the morning this new field will be initialized.
2. If this new field is initialized to use the new cmi constant QuoteUpdateControlValues. CONTROL_DISABLED, using this new cmi method will have the exact same behavior as the existing old block quote method, i.e. no protection against the quotes when such race condition rises.
3. When a QRM event or quote cancellation occurs, either user or system initiated, CBOEdirect will look at the value in the control field. It will not accept any more quotes from the user as long as the value remains the same as it was before the cancel event.
4. Once the user changes the value on a new quote, that quote will be accepted and that value becomes the new test condition.

An extra protection of quotes from being in the market after the logout will be provided in this release. When CBOEdirect processes the user logout event, if there is any in flight quotes coming into the system, CBOEdirect will reject those quotes or send a cancel report for those quotes. If the quotes are rejected, the returned quote result will contain the AuthorizationCodes. INVALID_SESSION_ID. If the quote cancel report is sent, the cancel reason will be specified as ActivityReasons. **INVALID_SESSION_ID.**

A new status message has been added that will be sent to a market maker if his quote or ICM order is part of a quote trigger. The trigger status will be sent out through the existing order status and quote status callbacks. New update reasons QUOTE_TRIGGER_BUY and QUOTE_TRIGGER_SELL have been added in the cmiConstants.idl.

Finally, a new cancelAllQuoteV3 method has been added. The parameter of this method remains the same as the old one. The difference in this new method is proper return of the DataValidationException when the trading session name that is entered is not valid.

CMi Quote

```
module cmiQuote

  typedef short QuoteUpdateControl;

   struct QuoteEntryStructV3
   {
      cmiQuote::QuoteEntryStruct quoteEntry;
      cmiQuote::QuoteUpdateControl quoteUpdateControlId;
   };

   struct QuoteStructV3
   {
      cmiQuote::QuoteStruct quote;
      cmiQuote::QuoteUpdateControl quoteUpdateControlId;
   };

   struct ClassQuoteResultStructV3
   {
      cmiQuote::ClassQuoteResultStructV2 quoteResult;
      cmiQuote::QuoteUpdateControl quoteUpdateControlId;
   };
   typedef sequence <ClassQuoteResultStructV3> ClassQuoteResultStructV3Sequence;
```

CMi V3

```
module cmiV3

interface Quote : cmiV2::Quote
  {
    cmiQuote::ClassQuoteResultStructV3Sequence acceptQuotesForClassV3(
      in cmiProduct::ClassKey classKey,
      in cmiQuote::QuoteEntryStructV3Sequence quotes )
        raises(
          exceptions::SystemException,
          exceptions::CommunicationException,
          exceptions::AuthorizationException,
          exceptions::DataValidationException,
          exceptions::NotAcceptedException,
          exceptions::TransactionFailedException
        );

    void cancelAllQuotesV3(
      in cmiSession::TradingSessionName sessionName)
        raises(
          exceptions::SystemException,
          exceptions::CommunicationException,
          exceptions::AuthorizationException,
          exceptions::DataValidationException,
          exceptions::NotAcceptedException,
          exceptions::TransactionFailedException
        );
  };
```

## 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 UserSessionManagerV3. This reference can be obtained through logon using the UserAccessV3 interface. The new logon method takes exactly the same parameters as the old CMi's logon method and returns a newly enhanced SessionManagerStructV3. The UserAccessV3 object will be made available as an alternate IOR link on the HTTP port that the CAS is publishing on.

CMi V3

```
module cmiV3

    interface UserAccessV3
    {
        UserSessionManagerV3 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
                );
    };

    interface UserSessionManagerV3 : cmiV2::UserSessionManagerV2, cmi::UserSessionManager
    {
        cmiV3::MarketQuery getMarketQueryV3()
            raises(
                exceptions::SystemException,
                exceptions::CommunicationException,
                exceptions::AuthorizationException
            );

        cmiV3::Quote getQuoteV3()
            raises(
                exceptions::SystemException,
                exceptions::CommunicationException,
                exceptions::AuthorizationException
            );
```

```
};
```

## IDL Interfaces

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

### cmiV3.idl

```
module cmiV3
{
  interface Quote : cmiV2::Quote

  {
    cmiQuote::ClassQuoteResultStructV3Sequence acceptQuotesForClassV3(
      in cmiProduct::ClassKey classKey,
      in cmiQuote::QuoteEntryStructV3Sequence quotes )
        raises(
          exceptions::SystemException,
          exceptions::CommunicationException,
          exceptions::AuthorizationException,
          exceptions::DataValidationException,
          exceptions::NotAcceptedException,
          exceptions::TransactionFailedException
        );

    void cancelAllQuotesV3(
      in cmiSession::TradingSessionName sessionName)
        raises(
          exceptions::SystemException,
          exceptions::CommunicationException,
          exceptions::AuthorizationException,
          exceptions::DataValidationException,
          exceptions::NotAcceptedException,
          exceptions::TransactionFailedException
        );
  };

  interface MarketQuery : cmiV2::MarketQuery
```

```
{
    cmiMarketData::MarketDataHistoryDetailStruct getDetailProductHistoryByTime(
        in string querySessionId,
            in cmiSession::TradingSessionName sessionName,
            in cmiProduct::ProductKey productKey,
            in cmiUtil::DateTimeStruct startTime,
            in cmiUtil::QueryDirection direction)
            raises(
                exceptions::SystemException,
                exceptions::CommunicationException,
                exceptions::DataValidationException,
                exceptions::NotFoundException,
                exceptions::AuthorizationException
            );

        cmiMarketData::MarketDataHistoryDetailStruct
getPriorityProductHistoryByTime(
        in string querySessionId,
            in cmiSession::TradingSessionName sessionName,
            in cmiProduct::ProductKey productKey,
            in cmiUtil::DateTimeStruct startTime,
            in cmiUtil::QueryDirection direction)
            raises(
                exceptions::SystemException,
                exceptions::CommunicationException,
                exceptions::DataValidationException,
                exceptions::NotFoundException,
                exceptions::AuthorizationException
            );

    void subscribeCurrentMarketForClassV3(
        in cmiSession::TradingSessionName sessionName,
        in cmiProduct::ClassKey classKey,
        in cmiCallbackV3::CMICurrentMarketConsumer clientListener,
        in cmiUtil::QueueAction actionOnQueue)
```

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

    void unsubscribeCurrentMarketForClassV3(
        in cmiSession::TradingSessionName sessionName,
        in cmiProduct::ClassKey classKey,
        in cmiCallbackV3::CMICurrentMarketConsumer clientListener)
            raises(
                exceptions::SystemException,
                exceptions::CommunicationException,
                exceptions::AuthorizationException,
                exceptions::DataValidationException
            );

    void subscribeCurrentMarketForProductV3(
        in cmiSession::TradingSessionName sessionName,
        in cmiProduct::ProductKey productKey,
        in cmiCallbackV3::CMICurrentMarketConsumer clientListener,
        in cmiUtil::QueueAction actionOnQueue)
            raises(
                exceptions::SystemException,
                exceptions::CommunicationException,
                exceptions::AuthorizationException,
                exceptions::DataValidationException
            );

    void unsubscribeCurrentMarketForProductV3(
        in cmiSession::TradingSessionName sessionName,
        in cmiProduct::ProductKey productKey,
        in cmiCallbackV3::CMICurrentMarketConsumer clientListener)
            raises(
                exceptions::SystemException,
```

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

    };

    interface UserSessionManagerV3 : cmiV2::UserSessionManagerV2,
cmi::UserSessionManager
    {
        cmiV3::MarketQuery getMarketQueryV3()
            raises(
                exceptions::SystemException,
                exceptions::CommunicationException,
                exceptions::AuthorizationException
            );

        cmiV3::Quote getQuoteV3()
            raises(
                exceptions::SystemException,
                exceptions::CommunicationException,
                exceptions::AuthorizationException
            );
    };

    interface UserAccessV3
    {
        UserSessionManagerV3 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
            );
    };
};
```

## cmiCallbackV3.idl

```
    module cmiCallbackV3
    {
      interface CMICurrentMarketConsumer {
        void acceptCurrentMarket(
          in cmiMarketData::CurrentMarketStructSequence bestMarkets,
              in cmiMarketData::CurrentMarketStructSequence bestPublicMarket,
          in long queueDepth,
          in cmiUtil::QueueAction queueAction);
      };
};
```

## cmiIntermarket.idl

```
interface IntermarketQuery
  {
    cmiIntermarketMessages::AdminStructSequence getAdminMessage(
      in cmiSession::TradingSessionName session,
      in cmiProduct::ProductKey productKey,
      in cmiAdmin::MessageKey adminMessageKey,
      in cmiUser::Exchange sourceExchange)
        raises(
          exceptions::SystemException,
          exceptions::CommunicationException,
          exceptions::AuthorizationException,
          exceptions::DataValidationException,
          exceptions::NotAcceptedException
        );
```

**cmiIntermarketMessages::BookDepthDetailedStruct showMarketableOrderBookAtPrice(**

**in cmiSession::TradingSessionName session,**

**in cmiProduct::ProductKey productKey,**

**in cmiUtil::PriceStruct openingPrice)**

**raises(**

**exceptions::SystemException,**

**exceptions::CommunicationException,**

**exceptions::AuthorizationException,**

**exceptions::DataValidationException,**

**exceptions::NotFoundException,**

**exceptions::NotAcceptedException**

);

*This method will get the orders on both the sell side and buy side for the given price. It will also return the buy orders greater then equal to and sell side less then equal to this price. It will also return any Market Orders that are in the OrderBook before the Opening. This call will be only available to the DPM.*

);

**cmiIntermarketMessages::OrderBookStatus getOrderBookStatus(**

**in cmiSession::TradingSessionName session,**

**in cmiProduct::ProductKey productKey)**

**raises(**

**exceptions::SystemException,**

**exceptions::CommunicationException,**

**exceptions::AuthorizationException,**

**exceptions::DataValidationException,**

**exceptions::NotFoundException,**

**exceptions::NotAcceptedException**

);

*This method will get the state of the OrderBook during the Product Opening state. This call will only available to the DPM*

}

## 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 short OrderBookPriceViewType;

   struct MarketVolumeStruct {
      cmiMarketData::VolumeType volumeType;
      long quantity;
      boolean multipleParties;
   };
   typedef sequence <MarketVolumeStruct> MarketVolumeStructSequence;

   struct CurrentMarketStruct {
      cmiProduct::ProductKeysStruct productKeys;
      cmiSession::TradingSessionName sessionName;
      string exchange;
      cmiUtil::PriceStruct bidPrice;
      cmiMarketData::MarketVolumeStructSequence bidSizeSequence;
      boolean bidIsMarketBest;
      cmiUtil::PriceStruct askPrice;
      cmiMarketData::MarketVolumeStructSequence askSizeSequence;
      boolean askIsMarketBest;
      cmiUtil::TimeStruct sentTime;
      boolean legalMarket;
   };
   typedef sequence <CurrentMarketStruct> CurrentMarketStructSequence;

   struct ExchangeVolumeStruct {
      string exchange;
      long volume;
   };
```

```
typedef sequence <ExchangeVolumeStruct> ExchangeVolumeStructSequence;

 struct NBBOStruct {
    cmiProduct::ProductKeysStruct productKeys;
    cmiSession::TradingSessionName sessionName;
    cmiUtil::PriceStruct bidPrice;
    cmiMarketData::ExchangeVolumeStructSequence bidExchangeVolume;
    cmiUtil::PriceStruct askPrice;
    cmiMarketData::ExchangeVolumeStructSequence askExchangeVolume;
    cmiUtil::TimeStruct sentTime;
};
typedef sequence <NBBOStruct> NBBOStructSequence;

struct RecapStruct
{
    cmiProduct::ProductKeysStruct productKeys;
    cmiSession::TradingSessionName sessionName;
    cmiProduct::ProductNameStruct productInformation;
    cmiUtil::PriceStruct lastSalePrice;
    cmiUtil::TimeStruct tradeTime;
    long lastSaleVolume;
    long totalVolume;
    char tickDirection; //Use the constants defined for TickDirectionType for this field.
    char netChangeDirection;
    char bidDirection;
    cmiUtil::PriceStruct netChange;
    cmiUtil::PriceStruct bidPrice;
    long bidSize;
    cmiUtil::TimeStruct bidTime;
    cmiUtil::PriceStruct askPrice;
    long askSize;
    cmiUtil::TimeStruct askTime;
    string recapPrefix;
    cmiUtil::PriceStruct tick;
    cmiUtil::PriceStruct lowPrice;
```

```
    cmiUtil::PriceStruct highPrice;
    cmiUtil::PriceStruct openPrice;
    cmiUtil::PriceStruct closePrice;
    long openInterest;
    cmiUtil::PriceStruct previousClosePrice;
    boolean isOTC;
};
typedef sequence < RecapStruct > RecapStructSequence;

struct TickerStruct
{
   cmiProduct::ProductKeysStruct productKeys;
   cmiSession::TradingSessionName sessionName;
   cmiProduct::Symbol exchangeSymbol;
   string salePrefix;
   cmiUtil::PriceStruct lastSalePrice;
   long lastSaleVolume;
   string salePostfix;
};
typedef sequence < TickerStruct > TickerStructSequence;

struct ExpectedOpeningPriceStruct
{
    cmiProduct::ProductKeysStruct productKeys;
    cmiSession::TradingSessionName sessionName;
    cmiMarketData::ExpectedOpeningPriceType eopType;
    cmiUtil::PriceStruct expectedOpeningPrice;
    long imbalanceQuantity;
    boolean legalMarket;
};
typedef sequence <ExpectedOpeningPriceStruct> ExpectedOpeningPriceStructSequence;

struct MarketDataHistoryEntryStruct {
    cmiMarketData::MarketDataHistoryEntryType entryType;
    cmiUtil::Source source;
    cmiUtil::DateTimeStruct reportTime;
```

```
    cmiUtil::PriceStruct price;
    long quantity;
    string sellerAcronym;
    string buyerAcronym;
    long bidSize;
    cmiUtil::PriceStruct bidPrice;
    long askSize;
    cmiUtil::PriceStruct askPrice;
    cmiUtil::PriceStruct underlyingLastSalePrice;
    cmiMarketData::ExpectedOpeningPriceType eopType;
    cmiSession::ProductState marketCondition;
    string optionalData;
    string exceptionCode;
    string physLocation;
    string prefix;
};
typedef sequence <MarketDataHistoryEntryStruct> MarketDataHistoryEntryStructSequence;

struct MarketDataHistoryStruct
{
    cmiProduct::ProductKeysStruct productKeys;
    cmiSession::TradingSessionName sessionName;
    cmiUtil::DateTimeStruct startTime;
    cmiUtil::DateTimeStruct endTime;
    cmiMarketData::MarketDataHistoryEntryStructSequence entries;
};
typedef sequence <MarketDataHistoryStruct> MarketDataHistoryStructSequence;

typedef short ExchangeIndicatorType ;

struct ExchangeIndicatorStruct
{
    string exchange;
    cmiMarketData::ExchangeIndicatorType marketCondition;
};
typedef sequence <ExchangeIndicatorStruct> ExchangeIndicatorStructSequence;
```

```
typedef char OverrideIndicatorType;

struct MarketDataDetailStruct
{
    cmiMarketData::OverrideIndicatorType overrideIndicator;
    cmiUtil::PriceStruct nbboAskPrice;
    cmiMarketData::ExchangeVolumeStructSequence nbboAskExchanges;
    cmiUtil::PriceStruct nbboBidPrice;
    cmiMarketData::ExchangeVolumeStructSequence nbboBidExchanges;
    boolean tradeThroughIndicator;
    cmiMarketData::ExchangeIndicatorStructSequence exchangeIndicators;
    cmiUtil::PriceStruct bestPublishedBidPrice;
    long bestPublishedBidVolume;
    cmiUtil::PriceStruct bestPublishedAskPrice;
    long bestPublishedAskVolume;
    cmiUser::ExchangeAcronymStructSequence brokers;
    cmiUser::ExchangeAcronymStructSequence contras;
    cmiUtil::KeyValueStructSequence extensions;
};

struct MarketDataHistoryDetailEntryStruct
{
    cmiMarketData::MarketDataHistoryEntryStruct historyEntry;
    cmiMarketData::MarketDataDetailStruct detailData;
};
typedef sequence <MarketDataHistoryDetailEntryStruct>
MarketDataHistoryDetailEntryStructSequence;

struct MarketDataHistoryDetailStruct
{
    cmiProduct::ProductKeysStruct productKeys;
    cmiSession::TradingSessionName sessionName;
    cmiUtil::DateTimeStruct startTime;
    cmiUtil::DateTimeStruct endTime;
    boolean isOutOfSequence;
```

```
    cmiMarketData::MarketDataHistoryDetailEntryStructSequence entries;
  };
  typedef sequence <MarketDataHistoryDetailStruct>
MarketDataHistoryDetailStructSequence;

  struct OrderBookPriceStruct
  {
    cmiUtil::PriceStruct price;
    long totalVolume;
    long contingencyVolume;
  };
  typedef sequence <OrderBookPriceStruct> OrderBookPriceStructSequence;

  struct BookDepthStruct
  {
    cmiProduct::ProductKeysStruct productKeys;
    cmiSession::TradingSessionName sessionName;
    cmiMarketData::OrderBookPriceStructSequence buySideSequence;
    cmiMarketData::OrderBookPriceStructSequence sellSideSequence;
    boolean allPricesIncluded;
    long transactionSequenceNumber;
  };
  typedef sequence <BookDepthStruct> BookDepthStructSequence;

  struct BookDepthUpdatePriceStruct {
    cmiMarketData::BookDepthUpdateType updateType;
    cmiUtil::PriceStruct price;
    long totalVolume;              // U: new quantity, I: quantity, D: ignored
    long contingencyVolume;        // U: new quantity, I: quantity, D: ignored
  };

  typedef sequence <BookDepthUpdatePriceStruct> BookDepthUpdatePriceStructSequence;

  struct BookDepthUpdateStruct {
    long sequenceNumber;
    cmiProduct::ProductKeysStruct productKeys;
```

```
    cmiSession::TradingSessionName sessionName;
    cmiMarketData::BookDepthUpdatePriceStructSequence buySideChanges;
    cmiMarketData::BookDepthUpdatePriceStructSequence sellSideChanges;
};

typedef sequence <BookDepthUpdateStruct> BookDepthUpdateStructSequence;

    struct CurrentMarketViewStruct
{
    cmiMarketData::CurrentMarketViewType currentMarketViewType;
    cmiUtil::PriceStruct bidPrice;
    cmiMarketData::MarketVolumeStructSequence bidSizeSequence;
    cmiUtil::PriceStruct askPrice;
    cmiMarketData::MarketVolumeStructSequence askSizeSequence;
};
typedef  sequence <CurrentMarketViewStruct> CurrentMarketViewStructSequence;

struct CurrentMarketStructV2
{
    cmiProduct::ProductKeysStruct productKeys;
    cmiSession::TradingSessionName sessionName;
    string exchange;
    cmiMarketData::CurrentMarketViewStructSequence currentMarketViews;
    cmiUtil::TimeStruct sentTime;
    boolean bidIsMarketBest;
    boolean askIsMarketBest;
    boolean legalMarket;
};
typedef sequence <CurrentMarketStructV2> CurrentMarketStructV2Sequence;

struct OrderBookPriceViewStruct
{
    cmiMarketData::OrderBookPriceViewType orderBookPriceViewType;
    cmiMarketData::MarketVolumeStructSequence viewSequence;
};
typedef sequence <OrderBookPriceViewStruct> OrderBookPriceViewStructSequence;
```

```
    struct OrderBookPriceStructV2
    {
        cmiUtil::PriceStruct price;
        OrderBookPriceViewStructSequence views;
    };
    typedef sequence <OrderBookPriceStructV2> OrderBookPriceStructV2Sequence;

    struct BookDepthStructV2
    {
        cmiProduct::ProductKeysStruct productKeys;
        cmiSession::TradingSessionName sessionName;
        cmiMarketData::OrderBookPriceStructV2Sequence buySideSequence;
        cmiMarketData::OrderBookPriceStructV2Sequence sellSideSequence;
        boolean allPricesIncluded;
        long transactionSequenceNumber;
    };

};
```

## cmiQuote. Idl

```
module cmiQuote
{
    typedef short RFQType;
    typedef long QuoteKey;
    typedef sequence <QuoteKey> QuoteKeySequence;
    typedef short QuoteUpdateControl;

    struct QuoteEntryStruct
    {
        cmiProduct::ProductKey productKey;
        cmiSession::TradingSessionName sessionName;
        cmiUtil::PriceStruct bidPrice;
        long bidQuantity;
        cmiUtil::PriceStruct askPrice;
        long askQuantity;
```

```
    string userAssignedId;
};
typedef sequence <QuoteEntryStruct> QuoteEntryStructSequence;

struct QuoteEntryStructV3
{
    cmiQuote::QuoteEntryStruct quoteEntry;
    cmiQuote::QuoteUpdateControl quoteUpdateControlId;
};

typedef sequence <QuoteEntryStructV3> QuoteEntryStructV3Sequence;

struct QuoteStruct
{
    cmiQuote::QuoteKey quoteKey;
    cmiProduct::ProductKey productKey;
    cmiSession::TradingSessionName sessionName;
    string userId;
    cmiUtil::PriceStruct bidPrice;
    long bidQuantity;
    cmiUtil::PriceStruct askPrice;
    long askQuantity;
    long transactionSequenceNumber;
    string userAssignedId;
};

typedef sequence <QuoteStruct> QuoteStructSequence;

struct QuoteStructV3
{
    cmiQuote::QuoteStruct quote;
    cmiQuote::QuoteUpdateControl quoteUpdateControlId;
};
typedef sequence <QuoteStructV3> QuoteStructV3Sequence;

struct QuoteDetailStruct
```

```
{
   cmiProduct::ProductKeysStruct productKeys;
   cmiProduct::ProductNameStruct productName;
   cmiUtil::UpdateStatusReason statusChange;
   cmiQuote::QuoteStruct quote;
};
typedef sequence <QuoteDetailStruct> QuoteDetailStructSequence;

struct RFQEntryStruct
{
   cmiProduct::ProductKey productKey;
   cmiSession::TradingSessionName sessionName;
   long quantity;
};


struct RFQStruct
{
   cmiProduct::ProductKeysStruct productKeys;
   cmiSession::TradingSessionName sessionName;
   long quantity;
   long timeToLive;
   cmiQuote::RFQType rfqType;
   cmiUtil::TimeStruct entryTime;
};
typedef sequence <RFQStruct> RFQStructSequence;

struct QuoteFilledReportStruct
{
   cmiQuote::QuoteKey quoteKey;
   cmiProduct::ProductKeysStruct productKeys;
   cmiProduct::ProductNameStruct productName;
   cmiOrder::FilledReportStructSequence filledReport;
   cmiUtil::UpdateStatusReason statusChange;
};
typedef sequence <QuoteFilledReportStruct> QuoteFilledReportStructSequence;
```

```
struct ClassQuoteResultStruct
{
    cmiProduct::ProductKey productKey;
    exceptions::ErrorCode errorCode;
};
typedef sequence <ClassQuoteResultStruct> ClassQuoteResultStructSequence;

struct ClassQuoteResultStructV2
{
    cmiQuote::QuoteKey quoteKey;
    cmiProduct::ProductKey productKey;
    exceptions::ErrorCode errorCode;
};
typedef sequence <ClassQuoteResultStructV2> ClassQuoteResultStructV2Sequence;

struct ClassQuoteResultStructV3
{
    cmiQuote::ClassQuoteResultStructV2 quoteResult;
    cmiQuote::QuoteUpdateControl quoteUpdateControlId;
};
typedef sequence <ClassQuoteResultStructV3> ClassQuoteResultStructV3Sequence;

struct QuoteRiskManagementProfileStruct
{
    cmiProduct::ClassKey classKey;
    long volumeThreshold;
    long timeWindow;
    boolean quoteRiskManagementEnabled;
};
typedef sequence <QuoteRiskManagementProfileStruct>
QuoteRiskManagementProfileStructSequence;

struct UserQuoteRiskManagementProfileStruct
{
    boolean globalQuoteRiskManagementEnabled;
```

```
    cmiQuote::QuoteRiskManagementProfileStruct defaultQuoteRiskProfile;
    cmiQuote::QuoteRiskManagementProfileStructSequence quoteRiskProfiles;
};
struct QuoteBustReportStruct
{
    cmiQuote::QuoteKey quoteKey;
    cmiProduct::ProductKeysStruct productKeys;
    cmiProduct::ProductNameStruct productName;
    cmiOrder::BustReportStructSequence bustedReport;
    cmiUtil::UpdateStatusReason statusChange;
};
typedef sequence <QuoteBustReportStruct> QuoteBustReportStructSequence;

struct QuoteCancelReportStruct
{
    cmiQuote::QuoteKey quoteKey;
    cmiProduct::ProductKeysStruct productKeys;
    cmiProduct::ProductNameStruct productName;
    cmiUtil::ActivityReason cancelReason;
    cmiUtil::UpdateStatusReason statusChange;
};
typedef sequence <QuoteCancelReportStruct> QuoteCancelReportStructSequence;

struct QuoteDeleteReportStruct {
    cmiQuote::QuoteDetailStruct quote;
    cmiUtil::ActivityReason deleteReason;
};
typedef sequence <QuoteDeleteReportStruct> QuoteDeleteReportStructSequence;

struct LockNotificationStruct {
    cmiSession::TradingSessionName sessionName;
    cmiProduct::ProductType productType;
    cmiProduct::ClassKey classKey;
    cmiProduct::ProductKey productKey;
    cmiUtil::Side side;
    cmiUtil::PriceStruct price;
```

```
        long quantity;
        string extensions;
        cmiUser::ExchangeAcronymStructSequence buySideUserAcronyms;
        cmiUser::ExchangeAcronymStructSequence sellSideUserAcronyms;
    };
    typedef sequence <LockNotificationStruct> LockNotificationStructSequence;

};
```

## cmiUser.idl

```
module cmiUser
{
    typedef char UserRole;
    typedef char LoginSessionMode;
    typedef string Exchange;
    typedef char  OriginType;
    typedef sequence <cmiUser::Exchange> ExchangeSequence;

    struct ExchangeFirmStruct
    {
        cmiUser::Exchange exchange;
        string firmNumber;
    };
    typedef sequence <ExchangeFirmStruct> ExchangeFirmStructSequence;

    struct ExchangeAcronymStruct
    {
        cmiUser::Exchange exchange;
        string acronym;
    };
    typedef sequence <ExchangeAcronymStruct> ExchangeAcronymStructSequence;

    struct PreferenceStruct
    {
        string name;
        string value;
```

```
};

typedef sequence <PreferenceStruct> PreferenceStructSequence;

struct ProfileStruct
{
   cmiProduct::ClassKey classKey;
   string account;
   string subAccount;
   cmiUser::ExchangeFirmStruct executingGiveupFirm;

};

typedef sequence <ProfileStruct> ProfileStructSequence;

struct SessionProfileStruct
{
   cmiProduct::ClassKey classKey;
   string account;
   string subAccount;
   cmiUser::ExchangeFirmStruct executingGiveupFirm;
        // session name of the profile, it applies all the sessions if it is ALL_SESSION_NAME
        cmiSession::TradingSessionName sessionName;
                // add one boolean to indicate if the account will be overwritten to blank account
        boolean isAccountBlanked;
   cmiUser::OriginType originCode;
};

typedef sequence <SessionProfileStruct> SessionProfileStructSequence;

struct AccountStruct
{
   string account;
   cmiUser::ExchangeFirmStruct executingGiveupFirm;
};
```

```
typedef sequence <AccountStruct> AccountStructSequence;

struct DpmStruct
{
   string dpmUserId;
   cmiProduct::ClassKeySequence dpmAssignedClasses;
};

typedef sequence <DpmStruct> DpmStructSequence;

struct UserStruct
{
   cmiUser::ExchangeAcronymStruct userAcronym;
   string userId; //unique per user
   cmiUser::ExchangeFirmStruct firm;
   string fullName;
   cmiUser::UserRole role;
   cmiUser::ExchangeFirmStructSequence executingGiveupFirms;
   cmiUser::ProfileStructSequence profilesByClass;
   cmiUser::ProfileStruct defaultProfile;
   cmiUser::AccountStructSequence accounts;
   cmiProduct::ClassKeySequence assignedClasses;
   cmiUser::DpmStructSequence dpms;
};
struct UserLogonStruct
{
   string userId; //unique per user
   string password;
   cmiUtil::VersionLabel version;
   cmiUser::LoginSessionMode loginMode;
};

struct SessionProfileUserStruct
{
      cmiUser::ExchangeAcronymStruct userAcronym;
      string userId; //unique per user
```

```
            cmiUser::ExchangeFirmStruct firm;
            string fullName;
            cmiUser::UserRole role;
            cmiUser::ExchangeFirmStructSequence executingGiveupFirms;
            cmiUser::AccountStructSequence accounts;
            cmiProduct::ClassKeySequence assignedClasses;
            cmiUser::DpmStructSequence dpms;
            cmiUser::SessionProfileStructSequence sessionProfilesByClass; // excludes all profiles
with default class key
            cmiUser::SessionProfileStructSequence defaultSessionProfiles;  // session default profiles
            cmiUser::SessionProfileStruct defaultProfile; //generic profile for all classes and all
sessions
    };

    typedef sequence <SessionProfileUserStruct> SessionProfileUserStructSequence;
};
```

## cmiConstants.idl

```
interface ExchangeIndicatorTypes
    {
        const cmiMarketData::ExchangeIndicatorType CLEAR = 21;
        const cmiMarketData::ExchangeIndicatorType HALTED = 22;
        const cmiMarketData::ExchangeIndicatorType FAST_MARKET = 23;
        const cmiMarketData::ExchangeIndicatorType OPENING_ROTATION = 24;
    };

    interface OverrideIndicatorTypes
    {
            const cmiMarketData::OverrideIndicatorType NONE = ' ';
            const cmiMarketData::OverrideIndicatorType LINKAGE = 'L';
            const cmiMarketData::OverrideIndicatorType BOOK_OVERRIDE = 'B';
            const cmiMarketData::OverrideIndicatorType OFFER_OVERRIDE = 'O';
            const cmiMarketData::OverrideIndicatorType SUPERVISORY_OVERRIDE =
'X';
    };
```

```
interface ExpectedOpeningPriceTypes
  {
    const cmiMarketData::ExpectedOpeningPriceType OPENING_PRICE = 1;
    const cmiMarketData::ExpectedOpeningPriceType MORE_BUYERS = 2;
    const cmiMarketData::ExpectedOpeningPriceType MORE_SELLERS = 3;
    const cmiMarketData::ExpectedOpeningPriceType NO_OPENING_TRADE = 4;
    const cmiMarketData::ExpectedOpeningPriceType MULTIPLE_OPENING_PRICES = 5;
    const cmiMarketData::ExpectedOpeningPriceType NEED_QUOTE_TO_OPEN = 6;
    const cmiMarketData::ExpectedOpeningPriceType PRICE_NOT_IN_QUOTE_RANGE = 7;
    const cmiMarketData::ExpectedOpeningPriceType NEED_DPM_QUOTE_TO_OPEN = 8;
    const cmiMarketData::ExpectedOpeningPriceType DPM_QUOTE_INVALID = 9;
  };
```

*This constant is used when the series can not open because of an invalid DPM quote.*

```
interface StatusUpdateReasons
  {
    const cmiUtil::UpdateStatusReason BOOKED = 1;
    const cmiUtil::UpdateStatusReason CANCEL = 2;
    const cmiUtil::UpdateStatusReason FILL = 3;
    const cmiUtil::UpdateStatusReason QUERY = 4;
    const cmiUtil::UpdateStatusReason UPDATE = 5;
    const cmiUtil::UpdateStatusReason OPEN_OUTCRY = 6;
    const cmiUtil::UpdateStatusReason NEW = 7;
    const cmiUtil::UpdateStatusReason BUST = 8;
    const cmiUtil::UpdateStatusReason REINSTATE = 9;
    const cmiUtil::UpdateStatusReason POSSIBLE_RESEND = 10;
    const cmiUtil::UpdateStatusReason QUOTE_TRIGGER_BUY = 11;
    const cmiUtil::UpdateStatusReason QUOTE_TRIGGER_SELL = 12;
  };

  interface ActivityFieldTypes
  {
    const cmiTraderActivity::ActivityFieldType    ORDERID        = 1;
    const cmiTraderActivity::ActivityFieldType    ACCOUNT        = 2;
    const cmiTraderActivity::ActivityFieldType    ASK_PRICE      = 3;
    const cmiTraderActivity::ActivityFieldType    ASK_QTY        = 4;
    const cmiTraderActivity::ActivityFieldType    BID_PRICE      = 5;
    const cmiTraderActivity::ActivityFieldType    BID_QTY        = 6;
```

```
    const cmiTraderActivity::ActivityFieldType    BUSTED_QUANTITY          = 7;
    const cmiTraderActivity::ActivityFieldType    CANCELLED_QUANTITY       = 8;
    const cmiTraderActivity::ActivityFieldType    CMTA                     = 9;
    const cmiTraderActivity::ActivityFieldType    CONTINGENCY_TYPE         = 10;
    const cmiTraderActivity::ActivityFieldType    EVENT_STATUS             = 11;   // Success
/ Failure
    const cmiTraderActivity::ActivityFieldType    LEAVES_QUANTITY          = 12;
    const cmiTraderActivity::ActivityFieldType    MISMATCHED_QUANTITY      = 13;
    const cmiTraderActivity::ActivityFieldType    OPTIONAL_DATA            = 14;
    const cmiTraderActivity::ActivityFieldType    ORIGINAL_QUANTITY        = 15;
    const cmiTraderActivity::ActivityFieldType    PRICE                    = 16;
    const cmiTraderActivity::ActivityFieldType    PRODUCT_STATE            = 17;   // to
capture FAST_MARKET
    const cmiTraderActivity::ActivityFieldType    QUANTITY                 = 18;
    const cmiTraderActivity::ActivityFieldType    QUOTEKEY                 = 19;
    const cmiTraderActivity::ActivityFieldType    REINSTATED_QUANTITY      = 20;
    const cmiTraderActivity::ActivityFieldType    REPLACE_ORDERID          = 21;
    const cmiTraderActivity::ActivityFieldType    RFQ_TYPE                 = 22;
    const cmiTraderActivity::ActivityFieldType    SIDE                     = 23;
    const cmiTraderActivity::ActivityFieldType    TIME_IN_FORCE            = 24;
    const cmiTraderActivity::ActivityFieldType    TIME_TO_LIVE             = 25;
    const cmiTraderActivity::ActivityFieldType    TLC_QUANTITY             = 26;
    const cmiTraderActivity::ActivityFieldType    TRADED_QUANTITY          = 27;
    const cmiTraderActivity::ActivityFieldType    TRADEID                  = 28;
    const cmiTraderActivity::ActivityFieldType    TRANSACTION_SEQUENCE_NUMBER =
29;
    const cmiTraderActivity::ActivityFieldType    USER_ASSIGNED_ID         = 30;
    const cmiTraderActivity::ActivityFieldType    CANCEL_REASON            = 31;
    const cmiTraderActivity::ActivityFieldType    BOOKED_QUANTITY          = 32;
    const cmiTraderActivity::ActivityFieldType    ORDER_STATE              = 33; // see
OrderStates
    const cmiTraderActivity::ActivityFieldType    PRODUCT                  = 34;
    const cmiTraderActivity::ActivityFieldType    EXEC_BROKER              = 35;
    const cmiTraderActivity::ActivityFieldType    QUOTE_UPDATE_CONTROL_ID  =
36;
  };
```

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

interface OrderBookStructTradableTypes
    {
            const cmiIntermarketMessages::OrderBookTradableType
        BOOK_ITEM_ORDER = 'O';
            const cmiIntermarketMessages::OrderBookTradableType
        BOOK_ITEM_QUOTE = 'Q'; // Quote Side
            const cmiIntermarketMessages::OrderBookTradableType
        BOOK_ITEM_QUOTE_TRIGGER = 'T';
    };
};
```

*The struct defines the different tradable types that can be returned as part of the book depth detail query. The struct that is returned as part of this query is "BookDepthDetailedStruct". This in turn contains the "OrderBookStruct". The OrderBookStruct returns the "tradableType". The User can use the values to change the display of the tradable as may be required.*

```
interface ActivityReasons
{
  const cmiUtil::ActivityReason NOTHING_DONE = 1;
  const cmiUtil::ActivityReason USER = 2;
  const cmiUtil::ActivityReason SYSTEM = 3;
  const cmiUtil::ActivityReason LOST_CONNECTION = 4;
  const cmiUtil::ActivityReason INSUFFICIENT_QUANTITY = 5;
  const cmiUtil::ActivityReason SPECIAL_ADJUSTMENT = 6;
```

const cmiUtil::ActivityReason QRM_REMOVED = 7;

const cmiUtil::ActivityReason INSUFFICIENT_QUANTITY_BUY_SIDE  = 8;

const cmiUtil::ActivityReason INSUFFICIENT_QUANTITY_SELL_SIDE  = 9;

**const cmiUtil::ActivityReason QUOTE_UPDATE_CONTROL =10;**

const cmiUtil::ActivityReason FAILOVER= 11;

**const cmiUtil::ActivityReason QUOTE_IN_TRIGGER =12;**

**const cmiUtil::ActivityReason INVALID_SESSION_ID =13;**

// The following are used for Linkage

const cmiUtil::ActivityReason BROKER_OPTION = 100;

const cmiUtil::ActivityReason CANCEL_PENDING = 101;

const cmiUtil::ActivityReason CROWD_TRADE = 102;

const cmiUtil::ActivityReason DUPLICATE_ORDER = 103;

const cmiUtil::ActivityReason EXCHANGE_CLOSED = 104;

const cmiUtil::ActivityReason GATE_VIOLATION = 105;

const cmiUtil::ActivityReason INVALID_ACCOUNT = 106;

const cmiUtil::ActivityReason INVALID_AUTOEX_VALUE = 107;

const cmiUtil::ActivityReason INVALID_CMTA = 108;

const cmiUtil::ActivityReason INVALID_FIRM = 109;

const cmiUtil::ActivityReason INVALID_ORIGIN_TYPE = 110;

const cmiUtil::ActivityReason INVALID_POSITION_EFFECT = 111;

const cmiUtil::ActivityReason INVALID_PRICE = 112;

const cmiUtil::ActivityReason INVALID_PRODUCT = 113;

const cmiUtil::ActivityReason INVALID_PRODUCT_TYPE = 114;

const cmiUtil::ActivityReason INVALID_QUANTITY = 115;

const cmiUtil::ActivityReason INVALID_SIDE = 116;

const cmiUtil::ActivityReason INVALID_SUBACCOUNT = 117;

const cmiUtil::ActivityReason INVALID_TIME_IN_FORCE = 118;

const cmiUtil::ActivityReason INVALID_USER = 119;

const cmiUtil::ActivityReason LATE_PRINT = 120;

const cmiUtil::ActivityReason NOT_FIRM = 121;

const cmiUtil::ActivityReason MISSING_EXEC_INFO = 122;

const cmiUtil::ActivityReason NO_MATCHING_ORDER = 123;

const cmiUtil::ActivityReason NON_BLOCK_TRADE = 124;

const cmiUtil::ActivityReason NOT_NBBO = 125;

```
    const cmiUtil::ActivityReason COMM_DELAYS = 126;
    const cmiUtil::ActivityReason ORIGINAL_ORDER_REJECTED = 127;
    const cmiUtil::ActivityReason OTHER = 128;
    const cmiUtil::ActivityReason PROCESSING_PROBLEMS = 129;
    const cmiUtil::ActivityReason PRODUCT_HALTED = 130;
    const cmiUtil::ActivityReason PRODUCT_IN_ROTATION = 131;
    const cmiUtil::ActivityReason STALE_EXECUTION = 132;
    const cmiUtil::ActivityReason STALE_ORDER = 133;
    const cmiUtil::ActivityReason ORDER_TOO_LATE = 134;
    const cmiUtil::ActivityReason TRADE_BUSTED = 135;
    const cmiUtil::ActivityReason TRADE_REJECTED = 136;
    const cmiUtil::ActivityReason ORDER_TIMEOUT = 141;
    const cmiUtil::ActivityReason REJECTED_LINKAGE_TRADE  = 170;
    const cmiUtil::ActivityReason SATISFACTION_ORD_REJ_OTHER  = 171;
    const cmiUtil::ActivityReason PRODUCT_SUSPENDED = 172;

    // Currently used for TPF linkage; in future may be used for CBOEdirect
    const cmiUtil::ActivityReason UNKNOWN_ORDER = 137;
    const cmiUtil::ActivityReason INVALD_EXCHANGE = 138;
    const cmiUtil::ActivityReason TRANSACTION_FAILED = 139;
    const cmiUtil::ActivityReason NOT_ACCEPTED = 140;

    // Used for linkage when cancel reason is not provided (could be user cancel or cancel
remaining)
    const cmiUtil::ActivityReason AWAY_EXCHANGE_CANCEL = 199;

    // Linkage Business Message Reject codes
    const cmiUtil::ActivityReason LINKAGE_CONDITIONAL_FIELD_MISSING = 900;
    const cmiUtil::ActivityReason LINKAGE_EXCHANGE_UNAVAILABLE = 901;
    const cmiUtil::ActivityReason LINKAGE_INVALID_MESSAGE = 902;
    const cmiUtil::ActivityReason LINKAGE_INVALID_DESTINATION = 903;
    const cmiUtil::ActivityReason LINKAGE_INVALID_PRODUCT = 904;
    const cmiUtil::ActivityReason LINKAGE_SESSION_REJECT = 905;
  };
```

**interface QuoteUpdateControlValues**

```
{
    const cmiQuote::QuoteUpdateControl CONTROL_DISABLED = 0;
};
```

## cmiUtil.idl

```
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 string Description;
   typedef long Key;
   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;
  };

  struct KeyValueStruct
  {
    string key;
    string value;
  };

  typedef sequence <KeyValueStruct> KeyValueStructSequence;

  struct KeyDescriptionStruct
  {
   cmiUtil::Key key;
   cmiUtil::Description description;
  };

  typedef sequence <KeyDescriptionStruct> KeyDescriptionStructSequence;
}
```

## CMi Error Codes

```
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;
  };
```

## CAS Simulator Changes

- Quote V3 example
- Current Market V3 example

## Document Changes

### API-01

- No changes.

### API-02

- CMi V3 examples for quotes and current market.
- CMi V3 functionality for quotes, market data and session management.
- If a user (one user ID, one user session) sends a number of mass quote or quote messages at the same time to the CAS for the same class, the CAS sends the quotes one at a time to the back-end server in the order the CAS receives them. If that user sends a bunch of mass quote or quote messages at the same time for different classes to the CAS, the CAS sends them as fast as possible to the back-end server. If the user sends a number of quote or mass quote packets all for the same class at exactly the same time, CBOE has no way to predict in what order the CAS will process them. Some firms have had a problem with this. CBOE requests that you not do this because the CAS might put them in an order that you don't want.

### API-03

- Added new interfaces based on this release.

### API-04

- Added new interfaces and definitions based on this release.

### API-05

- No changes

### API-06

- No changes

### API-07

- Corrected a few errors in the phase 1 requirements. In the cmi::Quote interface for market-makers and DPMs, getQuote is optional, cancelQuote and cancelQuotesByClass are required, and cancelAllQuotes is optional.

- Changed “RTH” to “W_MAIN”.

### CAS-01

- No changes.

### CAS-02

- New methods for the Intermarket Query interface.
  - ShowMarketableOrderBookAtPrice
  - GetOrderBookStatus
- New V3 methods for the Quote interface.
  - CancelAllQuotesV3
  - AcceptQuotesForClassV3
- New V3 methods for the Market Query interface.
- SubscribeCurrentMarketForClassV3
- SubscribeCurrentMarketForProductV3
- UnsbuscribeCurrentMarketForProductV3

## Test Plan Changes

### CMi Phase 2 Test Plans

- No changes

### 3a, Security Definition Test Plan

- Added a step to test the “Suspended” product state.

### 3b, Market Data Test Plan

- Changed “RTH” to “W_MAIN”.

### 3c, Quote Test Plan - Hybrid-ONE-CFE (includes 3j, Hybrid Sections and 3L, CFE Supplemental Tests)

- Changed “RTH” to “W_MAIN”.
- The permissible bid-ask differential in all Hybrid classes will be $5.00 regardless of premium value. Hybrid opening rotation quotes must meet non-Hybrid legal-width requirements. After the open, quotes may be $5 wide. This means that market makers cannot quote $5 wide during the open.

### 3d, RTH Order Test Plan

- Removed this document from the group of test plans.

### 3e, RTH-ONE-CFE Order Test Plan (includes 3k, CFE Supplemental Tests)

- Changed name of test plan from "RTH…" to "W_MAIN…" and changed "RTH" to "W_MAIN" throughout the document.

### 3f, Clearing Firm, Duplicate Message Test Plan

- Changed "RTH" to "W_MAIN".

### 3g, Strategy Quote Test Plan - ONE-CFE

- No changes

### 3h, RTH Strategy Order Test Plan

- Removed this document from the group of test plans.

### 3i, W_MAIN-ONE-CFE Strategy Order Test Plan

- Changed name of test plan from "RTH…" to "W_MAIN…" and changed "RTH" to "W_MAIN" throughout the document.

### 3m, Stock Trading On CBOEdirect (STOC) Order Test Plan

- No changes

### 3n, Stock Trading On CBOEdirect (STOC) Quote Test Plan

- No changes

### 3o, Stock Trading On CBOEdirect (STOC) DPM Administrative Test Plan

- No changes

### Phase 4 Test Plan

- No changes