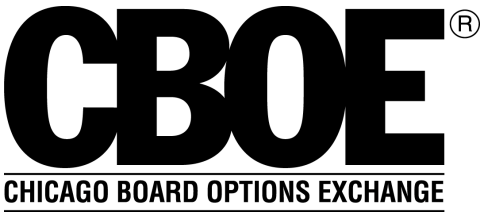

CBOE Application Programming Interface

CBOE API Version 2.0 Production - Release Notes

Provides an overview of the updates and changes to the CMi with this release.

CBOE PROPRIETARY INFORMATION

22 April 2002

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 1

INTRODUCTION 6

CMI RELEASE SCHEDULE 6
SUMMARY OF CHANGES IN CMI VERSION 2.0 6

SUMMARY OF CHANGES TO CMI IDL VERSION 2.0 PRODUCTION 1

CMI.IDL 1
MarketQuery *1*
Quote *1*
OrderQuery *1*
ProductQuery *1*
ProductDefinition *1*
OrderEntry *1*
UserPreferenceQuery *1*
Administrator *1*
TradingSession *1*
UserHistory *1*
UserTradingParameters *2*
UserSessionManager *2*
UserAccess *2*
CMICALLBACK.IDL 2
CMIADMIN.IDL 2
CMIMARKETDATA.IDL 2
CMIORDER.IDL 2
CMIPRODUCT.IDL 2
CMIQUOTE.IDL 2
CMISESSION.IDL 2
CMISTRATEGY.IDL 3
CMITRADERACTIVITY.IDL 3
CMIUSER.IDL 3
CMIUTIL.IDL 3
CMICONSTANTS.IDL 3
CMIERRORCODES.IDL 3

IDL CHANGES FOR THE PRODUCTION RELEASE 1

CMI.IDL 2
MarketQuery Interface *2*
Quote Interface *6*
OrderQuery Interface *6*
ProductQuery Interface *7*
ProductDefinition Interface *7*
OrderEntry Interface *7*
UserPreferenceQuery Interface *7*
Administrator Interface *7*
TradingSession Interface *7*

UserHistory Interface *7*
UserTradingParameters Interface *7*
UserSessionManager Interface *7*
UserAccess Interface *7*
CMICALLBACK.IDL 7
CMIADMIN.IDL 10
CMIMARKETDATA.IDL 10
CMIORDER.IDL 10
CMIPRODUCT.IDL 15
CMIQUOTE.IDL 18
CMISESSION.IDL 21
CMISTRATEGY.IDL 21
CMITRADERACTIVITY.IDL 21
CMIUSER.IDL 22
CMIUTIL.IDL 22
CMICONSTANTS.IDL 23
CMIERRORCODES.IDL 34

SUMMARY OF CHANGES TO CMI IDL BETA 2 37

CMI.IDL 37
MarketQuery *37*
Quote *37*
OrderQuery *37*
ProductQuery *37*
ProductDefinition *37*
OrderEntry *37*
UserPreferenceQuery *37*
Administrator *37*
TradingSession *37*
UserHistory *37*
UserTradingParameters *38*
UserSessionManager *38*
UserAccess *38*
CMICALLBACK.IDL 38
CMIADMIN.IDL 38
CMIMARKETDATA.IDL 38
CMIORDER.IDL 38
CMIPRODUCT.IDL 38
CMIQUOTE.IDL 38
CMISESSION.IDL 38
CMISTRATEGY.IDL 38
CMITRADERACTIVITY.IDL 39
CMIUSER.IDL 39
CMIUTIL.IDL 39
CMICONSTANTS.IDL 39
CMIERRORCODES.IDL 39

IDL CHANGES FOR BETA 2 40

CMI.IDL 41
MarketQuery Interface *41*
Quote Interface *41*
OrderQuery Interface *41*
ProductQuery Interface *41*
ProductDefinition Interface *41*

OrderEntry Interface ... *41*
UserPreferenceQuery Interface ... *41*
Administrator Interface ... *41*
TradingSession Interface ... *41*
UserHistory Interface ... *41*
UserTradingParameters Interface ... *41*
UserSessionManager Interface ... *41*
UserAccess Interface ... *42*
CMICALLBACK.IDL ... 42
CMIADMIN.IDL ... 42
CMIMARKETDATA.IDL ... 42
CMIORDER.IDL ... 42
CMIPRODUCT.IDL ... 47
CMIQUOTE.IDL ... 47
CMISESSION.IDL ... 47
CMISTRATEGY.IDL ... 48
CMITRADERACTIVITY.IDL ... 48
CMIUSER.IDL ... 48
CMIUTIL.IDL ... 49
CMICONSTANTS.IDL ... 49
CMIERRORCODES.IDL ... 60

SUMMARY OF CHANGES TO CMI IDL BETA 1 ... 63

CMI.IDL ... 63
MarketQuery ... *63*
Quote ... *63*
OrderQuery ... *63*
ProductQuery ... *63*
ProductDefinition ... *63*
OrderEntry ... *63*
UserPreferenceQuery ... *63*
Administrator ... *63*
TradingSession ... *63*
UserHistory ... *63*
UserTradingParameters ... *64*
UserSessionManager ... *64*
UserAccess ... *64*
CMICALLBACK.IDL ... 64
CMIADMIN.IDL ... 64
CMIMARKETDATA.IDL ... 64
CMIORDER.IDL ... 64
CMIPRODUCT.IDL ... 64
CMIQUOTE.IDL ... 64
CMISESSION.IDL ... 64
CMISTRATEGY.IDL ... 64
CMITRADERACTIVITY.IDL ... 65
CMIUSER.IDL ... 65
CMIUTIL.IDL ... 65
CMICONSTANTS.IDL ... 65
CMIERRORCODES.IDL ... 65

IDL CHANGES FOR BETA 1 ... 66

CMI.IDL ... 67
MarketQuery Interface ... *67*

Quote Interface ... *71*
OrderQuery Interface ... *72*
ProductQuery Interface ... *72*
ProductDefinition Interface ... *72*
OrderEntry Interface ... *72*
UserPreferenceQuery Interface ... *75*
Administrator Interface ... *75*
TradingSession Interface ... *75*
UserHistory Interface ... *75*
UserTradingParameters Interface ... *75*
UserSessionManager Interface ... *75*
UserAccess Interface ... *75*
CMICALLBACK.IDL ... 75
CMIADMIN.IDL ... 75
CMIMARKETDATA.IDL ... 76
CMIORDER.IDL ... 80
CMIPRODUCT.IDL ... 80
CMIQUOTE.IDL ... 80
CMISESSION.IDL ... 80
CMISTRATEGY.IDL ... 80
CMITRADERACTIVITY.IDL ... 80
CMIUSER.IDL ... 80
CMIUTIL.IDL ... 80
CMICONSTANTS.IDL ... 80
CMIERRORCODES.IDL ... 80

SUMMARY OF CHANGES TO CMI DOCUMENTATION ONLY ... 83

CMI.IDL ... 83
MarketQuery ... *83*
Quote ... *83*
OrderQuery ... *83*
ProductQuery ... *84*
ProductDefinition ... *84*
OrderEntry ... *84*
UserPreferenceQuery ... *84*
Administrator ... *84*
TradingSession ... *84*
UserHistory ... *85*
UserTradingParameters ... *85*
UserSessionManager ... *85*
UserAccess ... *85*
CMICALLBACK.IDL ... 85
CMIADMIN.IDL ... 85
CMIMARKETDATA.IDL ... 85
CMIORDER.IDL ... 85
CMIPRODUCT.IDL ... 86
CMIQUOTE.IDL ... 86
CMISESSION.IDL ... 86
CMISTRATEGY.IDL ... 86
CMITRADERACTIVITY.IDL ... 86
CMIUSER.IDL ... 86
CMIUTIL.IDL ... 86
CMICONSTANTS.IDL ... 86
CMIERRORCODES.IDL ... 86

IDL CHANGES FOR DOCUMENTATION ONLY 87

CMI.IDL 87
MarketQuery Interface 87
Quote Interface 91
OrderQuery Interface 95
ProductQuery Interface 98
ProductDefinition Interface 98
OrderEntry Interface 99
UserPreferenceQuery Interface 102
Administrator Interface 102
TradingSession Interface 102
UserHistory Interface 105
UserTradingParameters Interface 105
UserSessionManager Interface 105
UserAccess Interface 106
CMICALLBACK.IDL 106
CMIADMIN.IDL 108
CMIMARKETDATA.IDL 108
CMIORDER.IDL 112
CMIPRODUCT.IDL 118
CMIQUOTE.IDL 120
CMISESSION.IDL 122
CMISTRATEGY.IDL 124
CMITRADERACTIVITY.IDL 125
CMIUSER.IDL 125
CMIUTIL.IDL 126
CMICONSTANTS.IDL 127

CMICONSTANTS.IDLEXAMPLE PROGRAM CHANGES 137

JAVA EXAMPLES 137
Common 137
Example 1 137
Example 2 137
Example 3 137
Example 4 137
Example 5 137
Example 6 137
C++ EXAMPLES 138

DOCUMENTATION CHANGES 139

API-01 139
API-02 139
API-03 139
API-04 139
API-05 139
API-06 139
API-07 139
API-08 139

SIMULATOR CHANGES 140

Introduction

This document describes the changes that have been introduced to the CMi API. The IDL changes are detailed below for the release of :

- CBOEdirect Version 2.0 Production Release
- CBOEdirect Version 2.0b Software Development Kit Beta 2
- CBOEdirect Version 2.0a Software Development Kit Beta 1
- CBOEdirect Version 2.0 Documentation Only

CMi Release Schedule

Version	Description	Planned Date
2.0	CMi Version 2.0 Documentation Only	December 2001
2.0a	CMi Version 2.0 Software Development Kit Beta 1 Release	January 2002
2.0b	CMi Version 2.0 Software Development Kit Beta 2 Release	February 2002
2.0c	Production Version	April 2002

Summary of Changes in CMi Version 2.0

CBOEdirect Version 2.0 provides support for the following:

- Definition of and electronic trading of complex orders (option strategies).
- Dynamic Book Depth update
- Unique user identification across exchanges
- Guaranteed message delivery for order status, quote status, and text messages.
- Ability to specify at subscription time whether or not to receive publication of status on orders or quote. Prior to Version 2.0 users always received status on current orders and quote when logging on to CBOEdirect.
- Support for multiple logins for CBOEdirect

Summary of Changes to CMi IDL Version 2.0 Production

This section describes the changes in the IDL between the Production release and CMi Beta 2 release.

cmi.idl

MarketQuery

Added a NotAcceptedException to the getBookDepth method.

Quote

No change.

OrderQuery

No change.

ProductQuery

No change.

ProductDefinition

No change.

OrderEntry

No change.

UserPreferenceQuery

No change.

Administrator

No change.

TradingSession

No change.

UserHistory

No change.

UserTradingParameters

No change.

UserSessionManager

No change.

UserAccess

No change.

cmiCallBack.idl

Added a new method to the CMIQuoteStatusConsumer

- acceptQuoteCancelReport

cmiAdmin.idl

No change.

cmiMarketData.idl

No change.

cmiOrder.idl

cancelReportStruct

- added the cancelReason field
- removed the boolean nothingDone

cmiProduct.idl

Added the ReportingClassStruct

Added the ReportingClassStructSequence typedef sequence

cmiQuote.idl

Added the QuoteCancelReportStruct and typedef sequence.

cmiSession.idl

No change.

cmiStrategy.idl

No change.

cmiTraderActivity.idl

Removed the ActivityReason typedef

cmiUser.idl

No change.

cmiUtil.idl

Added the ActivityReason typedef.

cmiConstants.idl

Added new activity reasons:

- LOST_CONNECTION
- INSUFFICIENT_QUANTITY
- SPECIAL_ADJUSTMENT
- QRM_REMOVED

Added a new order state: Waiting

Added the OriginType: Market_Maker_Away

Added the PRODUCT_MAINTENANCE user role

Added a new exchange constant: NQLX

Added the BUY_WRITE StrategyType.

Removed the constants: Removed and QRM_Removed

cmiErrorCodes.idl

Added the error codes listed below:

- DataValidationCodes
  - INVALID_TRADE_TYPE
  - INVALID_TRADE_SOURCE
  - INVALID_USER
  - USER_NOT_ENABLED
  - NO_REMAINING_QUANTITY

Removed the error codes listed below:

- TransactionFailedCodes
  - INSUFFICIENT_QUANTITY
  - REMAINING_QUANTITY_MISMATCH
  - NO_REMAINING_QUANTITY

IDL Changes for the Production Release

IDL Changes are shown as follows:

Code that is unchanged from the previous version is shown in regular text:

```
interface MarketQuery
    {
        void subscribeRecapForClass(
            in cmiSession::TradingSessionName sessionName,
            in cmiProduct::ClassKey classKey,
            in cmiCallback::CMIRecapConsumer clientListener)
                raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::DataValidationException
                );
```

IDL that has been added is shown in red inside a text box:

```
void aNewMethodName(
            in cmiSession::TradingSessionName sessionName,
            in cmiProduct::ProductKey productKey,
            in cmiCallback::CMIOrderBookConsumer
orderBookConsumer)
                raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::DataValidationException,
                    exceptions::NotFoundException
            );
```

IDL that has been deleted is shown in grey in strike through inside a text box:

~~void someDeletedOperation(~~
~~in cmiSession::TradingSessionName sessionName,~~
~~in cmiProduct::ProductKey productKey,~~
~~in cmiCallback::CMIOrderBookConsumer~~
~~orderBookConsumer)~~
~~raises(~~
~~exceptions::SystemException,~~
~~exceptions::CommunicationException,~~
~~exceptions::AuthorizationException,~~
~~exceptions::DataValidationException,~~
~~exceptions::NotFoundException~~

cmi.idl

MarketQuery Interface

```
interface MarketQuery
    {
        void subscribeRecapForClass(
            in cmiSession::TradingSessionName sessionName,
            in cmiProduct::ClassKey classKey,
            in cmiCallback::CMIRecapConsumer clientListener)
                raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::DataValidationException
                );

        void subscribeRecapForProduct(
            in cmiSession::TradingSessionName sessionName,
            in cmiProduct::ProductKey productKey,
            in cmiCallback::CMIRecapConsumer clientListener)
                raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::DataValidationException
                );

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

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

```
                    void subscribeNBBOForClass(
                       in cmiSession::TradingSessionName sessionName,
                       in cmiProduct::ClassKey classKey,
                       in cmiCallback::CMINBBOConsumer clientListener)
                           raises(
                               exceptions::SystemException,
                               exceptions::CommunicationException,
                               exceptions::AuthorizationException,
                               exceptions::DataValidationException
                           );

                   void subscribeNBBOForProduct(
                       in cmiSession::TradingSessionName sessionName,
                       in cmiProduct::ProductKey productKey,
                       in cmiCallback::CMINBBOConsumer clientListener)
                           raises(
                               exceptions::SystemException,
                               exceptions::CommunicationException,
                               exceptions::AuthorizationException,
                               exceptions::DataValidationException
                           );

                   void unsubscribeCurrentMarketForProduct(
                       in cmiSession::TradingSessionName sessionName,
                       in cmiProduct::ProductKey productKey,
                       in cmiCallback::CMICurrentMarketConsumer
clientListener)
                           raises(
                               exceptions::SystemException,
                               exceptions::CommunicationException,
                               exceptions::AuthorizationException,
                               exceptions::DataValidationException
                           );

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

                    void unsubscribeNBBOForProduct(
                       in cmiSession::TradingSessionName sessionName,
                       in cmiProduct::ProductKey productKey,
                       in cmiCallback::CMINBBOConsumer clientListener)
                           raises(
                               exceptions::SystemException,
                               exceptions::CommunicationException,
                               exceptions::AuthorizationException,
                               exceptions::DataValidationException
```

```
        );

    void unsubscribeNBBOForClass(
        in cmiSession::TradingSessionName sessionName,
        in cmiProduct::ClassKey classKey,
        in cmiCallback::CMINBBOConsumer clientListener)
            raises(
                exceptions::SystemException,
                exceptions::CommunicationException,
                exceptions::AuthorizationException,
                exceptions::DataValidationException
            );

    void subscribeTicker(
        in cmiSession::TradingSessionName sessionName,
        in cmiProduct::ProductKey productKey,
        in cmiCallback::CMITickerConsumer clientListener)
            raises(
                exceptions::SystemException,
                exceptions::CommunicationException,
                exceptions::AuthorizationException,
                exceptions::DataValidationException
            );

    void unsubscribeRecapForProduct(
        in cmiSession::TradingSessionName sessionName,
        in cmiProduct::ProductKey productKey,
        in cmiCallback::CMIRecapConsumer clientListener)
            raises(
                exceptions::SystemException,
                exceptions::CommunicationException,
                exceptions::AuthorizationException,
                exceptions::DataValidationException
            );

    void unsubscribeRecapForClass(
        in cmiSession::TradingSessionName sessionName,
        in cmiProduct::ClassKey classKey,
        in cmiCallback::CMIRecapConsumer clientListener)
            raises(
                exceptions::SystemException,
                exceptions::CommunicationException,
                exceptions::AuthorizationException,
                exceptions::DataValidationException
            );

    void unsubscribeTicker(
        in cmiSession::TradingSessionName sessionName,
        in cmiProduct::ProductKey productKey,
        in cmiCallback::CMITickerConsumer clientListener)
            raises(
                exceptions::SystemException,
                exceptions::CommunicationException,
                exceptions::AuthorizationException,
                exceptions::DataValidationException
```

```
                    );

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

            void subscribeExpectedOpeningPrice(
                in cmiSession::TradingSessionName sessionName,
                in cmiProduct::ClassKey classKey,
                in cmiCallback::CMIExpectedOpeningPriceConsumer
    clientListener)
                    raises(
                        exceptions::SystemException,
                        exceptions::CommunicationException,
                        exceptions::AuthorizationException,
                        exceptions::DataValidationException
                    );

            void unsubscribeExpectedOpeningPrice(
                in cmiSession::TradingSessionName sessionName,
                in cmiProduct::ClassKey classKey,
                in cmiCallback::CMIExpectedOpeningPriceConsumer
    clientListener)
                    raises(
                        exceptions::SystemException,
                        exceptions::CommunicationException,
                        exceptions::AuthorizationException,
                        exceptions::DataValidationException
                    );

            cmiMarketData::BookDepthStruct getBookDepth(
                in cmiSession::TradingSessionName sessionName,
                in cmiProduct::ProductKey productKey)
                    raises(
                        exceptions::SystemException,
                        exceptions::CommunicationException,
                        exceptions::AuthorizationException,
                        exceptions::DataValidationException,
                        exceptions::NotFoundException,
                        exceptions::NotAcceptedException
                    );


            void subscribeBookDepth(
                in cmiSession::TradingSessionName sessionName,
```

```
                    in cmiProduct::ProductKey productKey,
                    in cmiCallback::CMIOrderBookConsumer
orderBookConsumer)
                        raises(
                            exceptions::SystemException,
                            exceptions::CommunicationException,
                            exceptions::AuthorizationException,
                            exceptions::DataValidationException
                    );

                void unsubscribeBookDepth(
                    in cmiSession::TradingSessionName sessionName,
                    in cmiProduct::ProductKey productKey,
                    in cmiCallback::CMIOrderBookConsumer
orderBookConsumer)
                        raises(
                            exceptions::SystemException,
                            exceptions::CommunicationException,
                            exceptions::AuthorizationException,
                            exceptions::DataValidationException
                        );

                void subscribeBookDepthUpdate(
                    in cmiSession::TradingSessionName sessionName,
                    in cmiProduct::ProductKey productKey,
                    in cmiCallback::CMIOrderBookUpdateConsumer
orderBookConsumer)
                        raises(
                            exceptions::SystemException,
                            exceptions::CommunicationException,
                            exceptions::AuthorizationException,
                            exceptions::DataValidationException
                        );

                void unsubscribeBookDepthUpdate(
                    in cmiSession::TradingSessionName sessionName,
                    in cmiProduct::ProductKey productKey,
                    in cmiCallback::CMIOrderBookUpdateConsumer
orderBookConsumer)
                        raises(
                            exceptions::SystemException,
                            exceptions::CommunicationException,
                            exceptions::AuthorizationException,
                            exceptions::DataValidationException
                        );
            };
```

Quote Interface

No change between Beta 2 and the Production release.

OrderQuery Interface

No change between Beta 2 and the Production release.

ProductQuery Interface

No change between Beta 2 and the Production release.

ProductDefinition Interface

No change between Beta 2 and the Production release.

OrderEntry Interface

No change between Beta 2 and the Production release.

UserPreferenceQuery Interface

No change between Beta 2 and the Production release.

Administrator Interface

No change between Beta 2 and the Production release.

TradingSession Interface

No change between Beta 2 and the Production release.

UserHistory Interface

No change between Beta 2 and the Production release.

UserTradingParameters Interface

No change between Beta 2 and the Production release.

UserSessionManager Interface

No change between Beta 2 and the Production release.

UserAccess Interface

No change between Beta 2 and the Production release.

cmiCallback.idl

```
interface CMICurrentMarketConsumer {
        void acceptCurrentMarket(
            in cmiMarketData::CurrentMarketStructSequence
currentMarket);
    };

    interface CMINBBOConsumer {
      void acceptNBBO( in cmiMarketData::NBBOStructSequence
nbbo);
```

```
    };

    interface CMIExpectedOpeningPriceConsumer {
        void acceptExpectedOpeningPrice(
            in cmiMarketData::ExpectedOpeningPriceStruct
expectedOpeningPrice );
    };

    interface CMIOrderStatusConsumer {
        void acceptOrderStatus( in
cmiOrder::OrderDetailStructSequence orders);

        void acceptOrderCanceledReport(
            in cmiOrder::OrderCancelReportStruct canceledReport
);

        void acceptOrderFilledReport(
            in cmiOrder::OrderFilledReportStruct filledReport );

        void acceptOrderBustReport(
            in cmiOrder::OrderBustReportStruct bustReport);

        void acceptOrderBustReinstateReport(
            in cmiOrder::OrderBustReinstateReportStruct
bustReinstatedReport);

        void acceptNewOrder(in cmiOrder::OrderDetailStruct
order);
    };

    interface CMIQuoteStatusConsumer {
        void acceptQuoteStatus( in
cmiQuote::QuoteDetailStructSequence quotes);

        void acceptQuoteFilledReport(
            in cmiQuote::QuoteFilledReportStruct filledReport );

        void acceptQuoteBustReport(
            in cmiQuote::QuoteBustReportStruct bustReport );
        void acceptQuoteCancelReport(
            in cmiQuote::QuoteCancelReportStruct cancelReport );

};

    interface CMIRFQConsumer {
        void acceptRFQ(
            in cmiQuote::RFQStruct rfq);
    };

    interface CMIClassStatusConsumer {
        void updateProductClass(in cmiSession::SessionClassStruct
updatedClass);

        void acceptClassState(in
cmiSession::ClassStateStructSequence classState);
```

```
    };


    interface CMIStrategyStatusConsumer {
        void updateProductStrategy(in
cmiSession::SessionStrategyStructSequence updatedStrategies);
    };

    interface CMIProductStatusConsumer {
        void updateProduct(in cmiSession::SessionProductStruct
updatedProduct);
        void acceptProductState(in
cmiSession::ProductStateStructSequence productState);
    };

    interface CMITradingSessionStatusConsumer {
        void acceptTradingSessionState( in
cmiSession::TradingSessionStateStruct sessionState);
    };

    interface CMIRecapConsumer {
        void acceptRecap(
             in cmiMarketData::RecapStructSequence recap);
    };

    interface CMITickerConsumer {
        void acceptTicker(
             in cmiMarketData::TickerStructSequence ticker);
    };

    interface CMIUserSessionAdmin {
         cmiAdmin::HeartBeatStruct acceptHeartBeat (in
cmiAdmin::HeartBeatStruct heartbeat);
         void acceptLogout( in string reason );
         void acceptTextMessage( in cmiAdmin::MessageStruct
message );
         void acceptAuthenticationNotice();
         void acceptCallbackRemoval(    in
cmiUtil::CallbackInformationStruct callbackInformation,
                                        in string reason,
                                        in exceptions::ErrorCode
errorCode);

    };

    interface CMIOrderBookConsumer {
         void acceptBookDepth(in cmiMarketData::BookDepthStruct
productBook);
    };

    interface CMIOrderBookUpdateConsumer {
      void acceptBookDepthUpdate(in
cmiMarketData::BookDepthUpdateStruct productBook );
    };
};
```

cmiAdmin.idl

No change between Beta 2 and the Production release.

cmiMarketData.idl

No change between Beta 2 and the Production release.

cmiOrder.idl

```
module cmiOrder
{
    typedef short ContingencyType;
    typedef short OrderState;
    typedef char  TimeInForce;
    typedef char  PositionEffect;
    typedef char  OriginType;
    typedef char  Coverage;
    typedef boolean  CrossingIndicator;
    typedef short CancelType;
    typedef short NBBOProtectionType;


    struct OrderContingencyStruct
    {
        cmiOrder::ContingencyType type;
        cmiUtil::PriceStruct price;
        long volume;
    };

    struct OrderIdStruct
    {
        cmiUser::ExchangeFirmStruct executingOrGiveUpFirm;
        string branch;
        long branchSequenceNumber;
        string correspondentFirm;
        string orderDate; // YYYYMMDD format
        long highCboeId;
        long lowCboeId;
    };

    struct OrderEntryStruct
    {
        cmiUser::ExchangeFirmStruct executingOrGiveUpFirm;
        string branch;
        long branchSequenceNumber;
        string correspondentFirm;
        string orderDate; // YYYYMMDD format

        cmiUser::ExchangeAcronymStruct originator;
        long originalQuantity;
```

```
            cmiProduct::ProductKey productKey;
            cmiUtil::Side side;
            cmiUtil::PriceStruct price;
            cmiOrder::TimeInForce timeInForce;
            cmiUtil::DateTimeStruct expireTime;
            cmiOrder::OrderContingencyStruct contingency;
            cmiUser::ExchangeFirmStruct cmta;
            string extensions;
            string account;
            string subaccount;
            cmiOrder::PositionEffect positionEffect;
            cmiOrder::CrossingIndicator cross;
            cmiOrder::OriginType orderOriginType;
            cmiOrder::Coverage coverage;
            cmiOrder::NBBOProtectionType orderNBBOProtectionType;
            string optionalData;
            string userAssignedId;
            cmiSession::TradingSessionNameSequence sessionNames;
        };
        typedef sequence <OrderEntryStruct> OrderEntryStructSequence;


        struct LegOrderEntryStruct
        {
            cmiProduct::ProductKey productKey;
            cmiUtil::PriceStruct mustUsePrice;
            cmiUser::ExchangeFirmStruct clearingFirm;
            cmiOrder::Coverage coverage;
            cmiOrder::PositionEffect positionEffect;
        };
        typedef sequence <LegOrderEntryStruct>
    LegOrderEntryStructSequence;


        struct LegOrderDetailStruct
        {
            cmiProduct::ProductKey productKey;
            cmiUtil::PriceStruct mustUsePrice;
            cmiUser::ExchangeFirmStruct clearingFirm;
            cmiOrder::Coverage coverage;
            cmiOrder::PositionEffect positionEffect;
            cmiUtil::Side side;
            long originalQuantity;
            long tradedQuantity;
            long cancelledQuantity;
            long leavesQuantity;
        };
        typedef sequence <LegOrderDetailStruct>
    LegOrderDetailStructSequence;


        struct OrderStruct
        {
            OrderIdStruct orderId;
            cmiUser::ExchangeAcronymStruct originator;
```

```
                    // Fields from the OrderEntryStruct
                    long originalQuantity;
                    cmiProduct::ProductKey productKey;
                    cmiUtil::Side side;
                    cmiUtil::PriceStruct price;
                    cmiOrder::TimeInForce timeInForce;
                    cmiUtil::DateTimeStruct expireTime;
                    cmiOrder::OrderContingencyStruct contingency;
                    cmiUser::ExchangeFirmStruct cmta;
                    string extensions;
                    string account;
                    string subaccount;
                    cmiOrder::PositionEffect positionEffect;
                    cmiOrder::CrossingIndicator cross;
                    cmiOrder::OriginType orderOriginType;
                    cmiOrder::Coverage coverage;
                    cmiOrder::NBBOProtectionType orderNBBOProtectionType;
                    string optionalData;

                    // Additional Order Fields
                    string userId;
                    cmiUser::ExchangeAcronymStruct userAcronym;
                    cmiProduct::ProductType productType;
                    cmiProduct::ClassKey classKey;
                    cmiUtil::DateTimeStruct receivedTime;
                    cmiOrder::OrderState state;
                    long tradedQuantity;
                    long cancelledQuantity;

                    long leavesQuantity;
                    cmiUtil::PriceStruct averagePrice;
                    long sessionTradedQuantity;
                    long sessionCancelledQuantity;
                    cmiUtil::PriceStruct sessionAveragePrice;

                    string orsId;
                    cmiUtil::Source source;
                    cmiOrder::OrderIdStruct crossedOrder;
                    long transactionSequenceNumber;
                    string userAssignedId;
                    cmiSession::TradingSessionNameSequence sessionNames;
                    cmiSession::TradingSessionName activeSession;
                    cmiOrder::LegOrderDetailStructSequence legOrderDetails;
                };
                typedef sequence <OrderStruct> OrderStructSequence;

                struct OrderDetailStruct
                {
                    cmiProduct::ProductNameStruct productInformation;
                    cmiUtil::UpdateStatusReason statusChange;
                    cmiOrder::OrderStruct orderStruct;
                };
                typedef sequence <OrderDetailStruct>
            OrderDetailStructSequence;
```

```
    struct CancelReportStruct
    {
        cmiOrder::OrderIdStruct orderId;
        cmiUtil::ReportType cancelReportType;
cmiUtil::ActivityReason cancelReason;

cmiProduct::ProductKey productKey;
        cmiSession::TradingSessionName sessionName;
        long cancelledQuantity;
        long tlcQuantity;
        long mismatchedQuantity;
        cmiUtil::DateTimeStruct timeSent;
        string orsId;
        long totalCancelledQuantity;
        boolean nothingDone;

long transactionSequenceNumber;
        string userAssignedCancelId;
    };
    typedef sequence <CancelReportStruct>
CancelReportStructSequence;

    struct CancelRequestStruct
    {
        cmiOrder::OrderIdStruct orderId;
        cmiSession::TradingSessionName sessionName;
        string userAssignedCancelId;
        cmiOrder::CancelType cancelType;
        long quantity;
    };

    struct ContraPartyStruct
    {
        cmiUser::ExchangeAcronymStruct user;
        cmiUser::ExchangeFirmStruct firm;
        long quantity;
    };
    typedef sequence <ContraPartyStruct>
ContraPartyStructSequence;


    struct FilledReportStruct
    {
        cmiUtil::CboeIdStruct tradeId;
        cmiUtil::ReportType fillReportType;
        cmiUser::ExchangeFirmStruct executingOrGiveUpFirm;
        string userId;
        cmiUser::ExchangeAcronymStruct  userAcronym;
        cmiProduct::ProductKey productKey;
        cmiSession::TradingSessionName sessionName;
        long tradedQuantity;
        long leavesQuantity;
        cmiUtil::PriceStruct price;
        cmiUtil::Side side;
```

```
            string orsId;
            string executingBroker;
            cmiUser::ExchangeFirmStruct cmta;
            string account;
            string subaccount;
            cmiUser::ExchangeAcronymStruct originator;
            string optionalData;
            string userAssignedId;
            string extensions;
            cmiOrder::ContraPartyStructSequence contraParties;
            cmiUtil::DateTimeStruct timeSent;
            cmiOrder::PositionEffect positionEffect;
            long transactionSequenceNumber;
        };
        typedef sequence <FilledReportStruct>
    FilledReportStructSequence;

        struct OrderFilledReportStruct
        {
            cmiOrder::OrderDetailStruct filledOrder;
            cmiOrder::FilledReportStructSequence filledReport;
        };
        typedef sequence <OrderFilledReportStruct>
    OrderFilledReportStructSequence;

        struct OrderCancelReportStruct
        {
            cmiOrder::OrderDetailStruct cancelledOrder;
            cmiOrder::CancelReportStructSequence cancelReport;
        };
        typedef sequence <OrderCancelReportStruct>
    OrderCancelReportStructSequence;

        struct PendingOrderStruct {
            cmiProduct::PendingNameStruct pendingProductName;
            cmiOrder::OrderStruct pendingOrder;
            cmiOrder::OrderStruct currentOrder;
        };
        typedef sequence <PendingOrderStruct>
    PendingOrderStructSequence;

        struct BustReportStruct
        {
            cmiUtil::CboeIdStruct tradeId;
            cmiUtil::ReportType bustReportType;
            cmiSession::TradingSessionName sessionName;
            cmiUser::ExchangeFirmStruct executingOrGiveUpFirm;
            string userId;
            cmiUser::ExchangeAcronymStruct  userAcronym;
            long bustedQuantity;
            cmiUtil::PriceStruct price;
            cmiProduct::ProductKey productKey;
            cmiUtil::Side side;
            cmiUtil::DateTimeStruct timeSent;
            long reinstateRequestedQuantity;
```

```
            long transactionSequenceNumber;
        };
        typedef sequence <BustReportStruct> BustReportStructSequence;

        struct OrderBustReportStruct
        {
            cmiOrder::OrderDetailStruct bustedOrder;
            cmiOrder::BustReportStructSequence bustedReport;
        };
        typedef sequence <OrderBustReportStruct>
    OrderBustReportStructSequence;

        struct BustReinstateReportStruct
        {
            cmiUtil::CboeIdStruct tradeId;
            long bustedQuantity;
            long reinstatedQuantity;
            long totalRemainingQuantity;
            cmiUtil::PriceStruct price;
            cmiProduct::ProductKey productKey;
            cmiSession::TradingSessionName sessionName;
            cmiUtil::Side side;
            cmiUtil::DateTimeStruct timeSent;
            long transactionSequenceNumber;
        };

        typedef sequence <BustReinstateReportStruct>
    BustReinstateReportStructSequence;

        struct OrderBustReinstateReportStruct
        {
            cmiOrder::OrderDetailStruct reinstatedOrder;
            cmiOrder::BustReinstateReportStruct bustReinstatedReport;
        };

        typedef sequence <OrderBustReinstateReportStruct>
    OrderBustReinstateReportStructSequence;
    };
```

cmiProduct.idl

```
    module cmiProduct
    {

        typedef char OptionType;
        typedef char ExpirationStyle;
        typedef long ProductKey;
        typedef sequence <ProductKey> ProductKeySequence;
        typedef long ReportingClassKey;
        typedef sequence <ReportingClassKey>
    ReportingClassKeySequence;
        typedef long ClassKey;
        typedef sequence <ClassKey> ClassKeySequence;
        typedef short ProductType;
        typedef sequence <ProductType> ProductTypeSequence;
```

```
        typedef string Symbol;
        typedef sequence <Symbol> SymbolSequence;
        typedef double CommodityQuantity;
        typedef short ListingState;
        typedef sequence <ListingState> ListingStateSequence;
        typedef short PriceAdjustmentType;
        typedef short PriceAdjustmentAction;
        typedef short PriceDisplayType;
        typedef string ProductDescriptionName;

        struct ProductDescriptionStruct
        {
          cmiProduct::ProductDescriptionName  name;
          cmiProduct::ProductDescriptionName  baseDescriptionName;
          cmiUtil::PriceStruct  minimumStrikePriceFraction;
          cmiUtil::PriceStruct  maxStrikePrice;
          cmiUtil::PriceStruct  premiumBreakPoint;
          cmiUtil::PriceStruct  minimumAbovePremiumFraction;
          cmiUtil::PriceStruct  minimumBelowPremiumFraction;
          cmiProduct::PriceDisplayType priceDisplayType;
          cmiProduct::PriceDisplayType premiumPriceFormat;
          cmiProduct::PriceDisplayType strikePriceFormat;
          cmiProduct::PriceDisplayType underlyingPriceFormat;
        };
        typedef sequence <ProductDescriptionStruct>
    ProductDescriptionStructSequence;

        struct ProductNameStruct
        {
          cmiProduct::Symbol reportingClass;
          cmiUtil::PriceStruct exercisePrice;
          cmiUtil::DateStruct expirationDate;
          cmiProduct::OptionType optionType;
          cmiProduct::Symbol productSymbol;
        };

        typedef sequence <ProductNameStruct>
    ProductNameStructSequence;

        struct ProductKeysStruct {
          cmiProduct::ProductKey productKey;
          cmiProduct::ClassKey classKey;
          cmiProduct::ProductType productType;
          cmiProduct::ReportingClassKey reportingClass;
        };

        struct ListingStateStruct
        {
          cmiProduct::ProductKeysStruct productKeys;
          cmiProduct::ListingState productState;
        };
        typedef sequence <ListingStateStruct>
    ListingStateStructSequence;

        struct ProductStruct
```

```
    {
      cmiProduct::ProductKeysStruct productKeys;
      cmiProduct::ProductNameStruct productName;
      cmiProduct::ListingState listingState;
      string description;
      string companyName;
      string unitMeasure;
      cmiProduct::CommodityQuantity standardQuantity;
      cmiUtil::DateStruct maturityDate;
      cmiUtil::DateStruct activationDate;
      cmiUtil::DateStruct inactivationDate;
      cmiUtil::DateTimeStruct createdTime;
      cmiUtil::DateTimeStruct lastModifiedTime;
      char opraMonthCode;
      char opraPriceCode;
    };
    typedef sequence <ProductStruct> ProductStructSequence;

    struct EPWStruct {
        double minimumBidRange;
        double maximumBidRange;
        double maximumAllowableSpread;
    };
    typedef sequence <EPWStruct> EPWStructSequence;
struct ReportingClassStruct
    {
      cmiProduct::ReportingClassKey classKey;
      cmiProduct::ProductType productType;
      cmiProduct::Symbol reportingClassSymbol;
      cmiProduct::Symbol productClassSymbol;
      cmiProduct::ClassKey productClassKey;
      cmiProduct::ListingState listingState;
      long contractSize;
      string transactionFeeCode;
      cmiUtil::DateStruct activationDate;
      cmiUtil::DateStruct inactivationDate;
      cmiUtil::DateTimeStruct createdTime;
      cmiUtil::DateTimeStruct lastModifiedTime;
    };
    typedef sequence <ReportingClassStruct>
ReportingClassStructSequence;

struct ClassStruct
    {
      cmiProduct::ClassKey classKey;
      cmiProduct::ProductType productType;
      cmiProduct::ListingState listingState;
      cmiProduct::Symbol classSymbol;
      cmiProduct::ProductStruct underlyingProduct;
      cmiProduct::Symbol primaryExchange;
      cmiUtil::DateStruct activationDate;
      cmiUtil::DateStruct inactivationDate;
      cmiUtil::DateTimeStruct createdTime;
      cmiUtil::DateTimeStruct lastModifiedTime;
      cmiProduct::EPWStructSequence epwValues;
```

```
      double epwFastMarketMultiplier;
      cmiProduct::ProductDescriptionStruct productDescription;
      boolean testClass;
      cmiProduct::ReportingClassStructSequence reportingClasses;
    };

    typedef sequence <ClassStruct> ClassStructSequence;

    struct ProductTypeStruct {
      cmiProduct::ProductType type;
      string name;
      string description;
      cmiUtil::DateTimeStruct createdTime;
      cmiUtil::DateTimeStruct lastModifiedTime;
    };

    typedef sequence <ProductTypeStruct>
ProductTypeStructSequence;

    struct PendingNameStruct
    {
      cmiProduct::PriceAdjustmentAction action;
      cmiProduct::ProductStruct productStruct;
      cmiProduct::ProductNameStruct pendingProductName;
    };
    typedef sequence <PendingNameStruct>
PendingNameStructSequence;

    struct PendingAdjustmentStruct
    {
      cmiProduct::ClassKey classKey;
      cmiUtil::DateStruct effectiveDate;
      cmiUtil::DateStruct submittedDate;
      cmiProduct::PriceAdjustmentType type;
      boolean active;
      cmiProduct::PendingNameStructSequence productsPending;
    };
    typedef sequence <PendingAdjustmentStruct>
PendingAdjustmentStructSequence;
};
```

cmiQuote.idl

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

    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 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 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 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;
```

```
};
```

cmiSession.idl

No change between Beta 2 and the Production release.

cmiStrategy.idl

No change between Beta 2 and the Production release.

cmiTraderActivity.idl

```
module cmiTraderActivity
{
    typedef short ActivityType;
    typedef short ActivityFieldType;
typedef short ActivityReason;


    struct ActivityFieldStruct
    {
        cmiTraderActivity::ActivityFieldType fieldType;
        string fieldName;
        string fieldValue;
    };
    typedef sequence <ActivityFieldStruct>
ActivityFieldStructSequence;

    struct ActivityRecordStruct
    {
        cmiProduct::ProductKey productKey;
        cmiUtil::DateTimeStruct eventTime;
        cmiTraderActivity::ActivityType entryType;
        cmiTraderActivity::ActivityFieldStructSequence
activityFields;
    };
    typedef sequence <ActivityRecordStruct>
ActivityRecordStructSequence;

    struct ActivityHistoryStruct
    {
        cmiProduct::ClassKey classKey;
        cmiUtil::DateTimeStruct startTime;
        cmiUtil::DateTimeStruct endTime;
        cmiTraderActivity::ActivityRecordStructSequence
activityRecords;
    };
    typedef sequence <ActivityHistoryStruct>
ActivityHistoryStructSequence;
};
```

cmiUser.idl

No change between Beta 2 and the Production release.

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;

    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;
        };

    };
```

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 LoginSessionTypes
        {
            const cmiSession::LoginSessionType PRIMARY = 1;
            const cmiSession::LoginSessionType SECONDARY = 2;
        };

        interface MarketDataHistoryEntryTypes
        {
            const cmiMarketData::MarketDataHistoryEntryType
    QUOTE_ENTRY = 1;
            const cmiMarketData::MarketDataHistoryEntryType
    PRICE_REPORT_ENTRY = 2;
            const cmiMarketData::MarketDataHistoryEntryType
    EXPECTED_OPEN_PRICE = 3;
            const cmiMarketData::MarketDataHistoryEntryType
    MARKET_CONDITION_ENTRY = 4;
            const cmiMarketData::MarketDataHistoryEntryType
    UNSIZED_QUOTE_ENTRY = 5;
        };

        interface MarketChangeReasons
        {
            const cmiMarketData::MarketChangeReason EXCHANGE = 1;
            const cmiMarketData::MarketChangeReason NBBO = 2;
```

```
    const cmiMarketData::MarketChangeReason COMBINED = 3;
};

interface QueryDirections
{
    const cmiUtil::QueryDirection QUERY_FORWARD = 1;
    const cmiUtil::QueryDirection QUERY_BACKWARD = 2;
};


interface PriceDisplayTypes
{
    const cmiProduct::PriceDisplayType FRACTION = 1;
    const cmiProduct::PriceDisplayType DECIMAL = 2;
};

interface OptionTypes
{
    const cmiProduct::OptionType CALL = 'C';
    const cmiProduct::OptionType PUT = 'P';
};

interface ProductTypes
{
    const cmiProduct::ProductType COMMODITY = 1;
    const cmiProduct::ProductType DEBT = 2;
    const cmiProduct::ProductType EQUITY = 3;
    const cmiProduct::ProductType FUTURE = 4;
    const cmiProduct::ProductType INDEX = 5;
    const cmiProduct::ProductType LINKED_NOTE = 6;
    const cmiProduct::ProductType OPTION = 7;
    const cmiProduct::ProductType UNIT_INVESTMENT_TRUST = 8;
    const cmiProduct::ProductType VOLATILITY_INDEX = 9;
    const cmiProduct::ProductType WARRANT = 10;
    const cmiProduct::ProductType STRATEGY = 11;
};

interface ProductStates
{
    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;
};

interface ListingStates
{
    const cmiProduct::ListingState ACTIVE = 1;
    const cmiProduct::ListingState INACTIVE = 2;
    const cmiProduct::ListingState UNLISTED = 3;
```

```
            const cmiProduct::ListingState OBSOLETE = 4;
        };

        interface ClassStates
        {
            const cmiSession::ClassState NOT_IMPLEMENTED = 1;
        };

        interface TradingSessionStates
        {
            const cmiSession::TradingSessionState CLOSED = 1;
            const cmiSession::TradingSessionState OPEN = 2;
        };

        interface TradingSessionType
        {
            const cmiSession::TradingSessionState DAY = 1;
            const cmiSession::TradingSessionState EVENING = 2;
        };

        interface TradingSessionMethod
        {
            const cmiSession::TradingSessionState SBT = 1;
            const cmiSession::TradingSessionState OPENOUTCRY = 2;
        };

        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 REMOVED = 11;
            const cmiUtil::UpdateStatusReason QRM_REMOVED = 12;

        };
    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
    };

    interface VolumeTypes
    {
        const cmiMarketData::VolumeType LIMIT = 1; // Limit (no
contingency)
        const cmiMarketData::VolumeType AON = 2; // All or None
        const cmiMarketData::VolumeType FOK = 3; // Fill or Kill
        const cmiMarketData::VolumeType IOC = 4; // Immediate or
Cancel
    };

    interface OrderStates
    {
        const cmiOrder::OrderState BOOKED = 1;
        const cmiOrder::OrderState CANCEL = 2;
        const cmiOrder::OrderState FILL = 3;
        const cmiOrder::OrderState OPEN_OUTCRY = 4;
        const cmiOrder::OrderState INACTIVE = 5;
        const cmiOrder::OrderState ACTIVE = 6;
        const cmiOrder::OrderState EXPIRED = 7;
        const cmiOrder::OrderState PURGED = 8;
        const cmiOrder::OrderState REMOVED = 9;
        const cmiOrder::OrderState WAITING = 10;

};

    interface RFQTypes
    {
        const cmiQuote::RFQType MANUAL = 1;
        const cmiQuote::RFQType SYSTEM = 2;
    };

    interface Sides
    {
        const cmiUtil::Side BUY  = 'B';
        const cmiUtil::Side SELL = 'S';
        const cmiUtil::Side BID  = 'B';
        const cmiUtil::Side ASK  = 'A';
        const cmiUtil::Side AS_DEFINED = 'D';
        const cmiUtil::Side OPPOSITE = 'O';
    };

    interface TimesInForce
    {
        const cmiOrder::TimeInForce GTC = 'G';
        const cmiOrder::TimeInForce DAY = 'D';
```

```
            const cmiOrder::TimeInForce GTD = 'T'; // Good until
        datetime
            };

            interface PositionEffects
            {
                const cmiOrder::PositionEffect OPEN = 'O';
                const cmiOrder::PositionEffect CLOSED = 'C';
                const cmiOrder::PositionEffect NOTAPPLICABLE = 'N';
            };

            interface OrderOrigins
            {
                const cmiOrder::OriginType CUSTOMER = 'C';
                const cmiOrder::OriginType FIRM = 'F';
                const cmiOrder::OriginType BROKER_DEALER = 'B';
                const cmiOrder::OriginType CUSTOMER_BROKER_DEALER = 'X';
                const cmiOrder::OriginType MARKET_MAKER = 'M';
                const cmiOrder::OriginType MARKET_MAKER_AWAY = 'N';
                const cmiOrder::OriginType CTI1Origin1 = 'V';
                const cmiOrder::OriginType CTI1Origin2 = 'E';
                const cmiOrder::OriginType CTI1Origin5 = 'Q';
                const cmiOrder::OriginType CTI3Origin1 = 'G';
                const cmiOrder::OriginType CTI3Origin2 = 'H';
                const cmiOrder::OriginType CTI3Origin5 = 'R';
                const cmiOrder::OriginType CTI4Origin2 = 'O';
                const cmiOrder::OriginType CTI4Origin5 = 'T';
             //Non Member, SIPC Protected Account
            };
        interface UserRoles
            {
                const cmiUser::UserRole FIRM = 'F';
                const cmiUser::UserRole BROKER_DEALER = 'B';
                const cmiUser::UserRole CUSTOMER_BROKER_DEALER = 'X';
                const cmiUser::UserRole MARKET_MAKER = 'M';
                const cmiUser::UserRole HELP_DESK = 'H';
                const cmiUser::UserRole DPM_ROLE = 'D';
                const cmiUser::UserRole UNKNOWN_ROLE = 'K';
                const cmiUser::UserRole CLASS_DISPLAY = 'C';
                const cmiUser::UserRole FIRM_DISPLAY = 'R';
                const cmiUser::UserRole EXCHANGE_BROKER = 'E';
                const cmiUser::UserRole PRODUCT_MAINTENANCE = 'P';
    };
};


            interface EntryTypes
            {
                const cmiUtil::EntryType ADD = 'A';
                const cmiUtil::EntryType CANCEL = 'C';
                const cmiUtil::EntryType CANCEL_REPLACE = 'R';
                const cmiUtil::EntryType FILL = 'F';
                const cmiUtil::EntryType BOOK = 'B';
                const cmiUtil::EntryType STATE = 'S';
                const cmiUtil::EntryType PRICE_ADJUST = 'P';
```

```
        const cmiUtil::EntryType UPDATE = 'U';
        const cmiUtil::EntryType BUST = 'K';
    };

    interface CoverageTypes
    {
        const cmiOrder::Coverage UNSPECIFIED = 'B';
        const cmiOrder::Coverage COVERED = 'C';
        const cmiOrder::Coverage UNCOVERED = 'U';
    };

    interface Sources
    {
        const cmiUtil::Source TPF = 'T';
        const cmiUtil::Source SBT = 'S';
        const cmiUtil::Source COMPASS = 'C';
    };

    interface PriceTypes
    {
        const cmiUtil::PriceType NO_PRICE = 1;
        const cmiUtil::PriceType LIMIT = 2;
        const cmiUtil::PriceType VALUED = 2;
        const cmiUtil::PriceType MARKET = 3;
    };

    interface ExpirationStyles
    {
        const cmiProduct::ExpirationStyle EUROPEAN = 'E';
        const cmiProduct::ExpirationStyle AMERICAN = 'A';
    };

    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
    };
};


    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;
    };

    interface PriceAdjustmentTypes
    {
        const cmiProduct::PriceAdjustmentType SPLIT = 1;
        const cmiProduct::PriceAdjustmentType DIVIDEND_CASH = 2;
        const cmiProduct::PriceAdjustmentType DIVIDEND_PERCENT =
3;
        const cmiProduct::PriceAdjustmentType DIVIDEND_STOCK = 4;
        const cmiProduct::PriceAdjustmentType LEAP_ROLLOVER = 5;
        const cmiProduct::PriceAdjustmentType MERGER = 6;
        const cmiProduct::PriceAdjustmentType SYMBOL_CHANGE = 7;
        const cmiProduct::PriceAdjustmentType COMMON_DISTRIBUTION
= 8;
    };


    interface PriceAdjustmentActions
    {
        const cmiProduct::PriceAdjustmentAction
PRICE_ADJUSTMENT_UPDATE = 1;
```

```
        const cmiProduct::PriceAdjustmentAction
PRICE_ADJUSTMENT_DELETE = 2;
        const cmiProduct::PriceAdjustmentAction
PRICE_ADJUSTMENT_CREATE = 3;
        const cmiProduct::PriceAdjustmentAction
PRICE_ADJUSTMENT_MOVE = 4;
    };

    interface PriceScale
    {
        const long DEFAULT_SCALE = 1000000000;
    };

    interface ProductClass
    {
        const long DEFAULT_CLASS_KEY = 0;
    };

    interface OrderCancelTypes
    {
        const cmiOrder::CancelType DESIRED_CANCEL_QUANTITY    =
1;
        const cmiOrder::CancelType DESIRED_REMAINING_QUANTITY =
2;
        const cmiOrder::CancelType CANCEL_ALL_QUANTITY        =
3;
    };


    interface ActivityTypes
    {
        // Order Activity Events
        const cmiTraderActivity::ActivityType  NEW_ORDER
= 1;
        const cmiTraderActivity::ActivityType  FILL_ORDER
= 2;
        const cmiTraderActivity::ActivityType  CANCEL_ORDER
= 3;
        const cmiTraderActivity::ActivityType  BUST_ORDER_FILL
= 4;
        const cmiTraderActivity::ActivityType
BUST_REINSTATE_ORDER    = 5;
        const cmiTraderActivity::ActivityType
CANCEL_REPLACE_ORDER    = 6;
        const cmiTraderActivity::ActivityType  UPDATE_ORDER
= 7;
        const cmiTraderActivity::ActivityType  BOOK_ORDER
= 8;

const cmiTraderActivity::ActivityType  STATE_CHANGE_ORDER      =
9;

const cmiTraderActivity::ActivityType  PRICE_ADJUST_ORDER      =
10;
```

```
        const cmiTraderActivity::ActivityType  CANCEL_ALL_ORDERS
= 11;

        // Strategy Order leg activity types
        const cmiTraderActivity::ActivityType
NEW_ORDER_STRATEGY_LEG      = 51;
        const cmiTraderActivity::ActivityType  FILL_STRATEGY_LEG
= 52;
        const cmiTraderActivity::ActivityType
CANCEL_STRATEGY_LEG         = 53;
        const cmiTraderActivity::ActivityType
BUST_STRATEGY_LEG_FILL      = 54;
        const cmiTraderActivity::ActivityType
BUST_REINSTATE_STRATEGY_LEG = 55;
        const cmiTraderActivity::ActivityType
UPDATE_STRATEGY_LEG         = 57;
        const cmiTraderActivity::ActivityType
PRICE_ADJUST_ORDER_LEG      = 60;

        // Quote Activity Events
        const cmiTraderActivity::ActivityType  NEW_QUOTE
= 101;
        const cmiTraderActivity::ActivityType  FILL_QUOTE
= 102;
        const cmiTraderActivity::ActivityType  CANCEL_QUOTE
= 103;
        const cmiTraderActivity::ActivityType  CANCEL_ALL_QUOTES
= 104;
        const cmiTraderActivity::ActivityType
SYSTEM_CANCEL_QUOTE     = 105;
        const cmiTraderActivity::ActivityType  UPDATE_QUOTE
= 106;
        const cmiTraderActivity::ActivityType  BUST_QUOTE_FILL
= 107;

        // Strategy Quote leg activity types
        const cmiTraderActivity::ActivityType  QUOTE_LEG_FILL
= 152;
        const cmiTraderActivity::ActivityType
BUST_QUOTE_LEG_FILL     = 157;

        // RFQ Activity Events
        const cmiTraderActivity::ActivityType  NEW_RFQ
= 201;

    };

    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;
    };

    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;
    };

    interface StrategyTypes
    {
        const cmiStrategy::StrategyType UNKNOWN = 1;
        const cmiStrategy::StrategyType STRADDLE = 2;
        const cmiStrategy::StrategyType PSEUDO_STRADDLE = 3;
        const cmiStrategy::StrategyType VERTICAL = 4;
        const cmiStrategy::StrategyType RATIO = 5;
        const cmiStrategy::StrategyType TIME = 6;
        const cmiStrategy::StrategyType DIAGONAL = 7;
        const cmiStrategy::StrategyType COMBO = 8;
        const cmiStrategy::StrategyType BUY_WRITE = 9;
    };

    interface ReportTypes
    {
        const cmiUtil::ReportType REGULAR_REPORT = 1;
        const cmiUtil::ReportType STRATEGY_REPORT = 2;
        const cmiUtil::ReportType STRATEGY_LEG_REPORT = 3;
    };

    interface BookDepthTypes
    {
        const cmiMarketData::BookDepthUpdateType DELETE_PRICE =
'D';        // delete book price
        const cmiMarketData::BookDepthUpdateType INSERT_PRICE =
'I';        // new book price
        const cmiMarketData::BookDepthUpdateType UPDATE_PRICE =
'U';        // book price update
    };

    interface OrderNBBOProtectionTypes
    {
```

```
        const cmiOrder::NBBOProtectionType NONE = 1;
        const cmiOrder::NBBOProtectionType FULL = 2;
    };
};
```

cmiErrorCodes.idl

```
module cmiErrorCodes
{
    interface DataValidationCodes  {
        const exceptions::ErrorCode INVALID_USER = 1080;
        const exceptions::ErrorCode DUPLICATE_ID = 1000;
        const exceptions::ErrorCode INVALID_VOLUME = 1010;

        const exceptions::ErrorCode INVALID_TIME = 1020;
        const exceptions::ErrorCode INCOMPLETE_QUOTE = 1030;
        const exceptions::ErrorCode INVALID_QUANTITY = 1040;
        const exceptions::ErrorCode INVALID_STRATEGY = 1060;
        const exceptions::ErrorCode INVALID_SPREAD = 1070;
        const exceptions::ErrorCode INVALID_MEMBERKEY = 1080;
        const exceptions::ErrorCode INVALID_PRODUCT = 1090;
        const exceptions::ErrorCode INVALID_SESSION = 1100;
        const exceptions::ErrorCode INVALID_STATE = 1110;
        const exceptions::ErrorCode PREFERENCE_PATH_MISMATCH =
1120;
        const exceptions::ErrorCode INVALID_ORDER_ID = 1130;
        const exceptions::ErrorCode LISTENER_ALREADY_REGISTERED =
1140;
        const exceptions::ErrorCode INVALID_SIDE = 1150;
        const exceptions::ErrorCode INVALID_PRICE = 1160;
        const exceptions::ErrorCode INVALID_UPDATE_ATTEMPT =
1170;
        const exceptions::ErrorCode INVALID_ORIGINATOR = 1180;
        const exceptions::ErrorCode INVALID_ACCOUNT = 1200;
        const exceptions::ErrorCode INVALID_EXECUTING_GIVEUP_FIRM
= 1210;
        const exceptions::ErrorCode INVALID_CONTINGENCY_TYPE =
1220;
        const exceptions::ErrorCode INVALID_TIME_IN_FORCE = 1230;
        const exceptions::ErrorCode INVALID_POSITION_EFFECT =
1240;
        const exceptions::ErrorCode INVALID_ORIGIN_TYPE = 1250;
        const exceptions::ErrorCode INVALID_COVERAGE = 1260;
        const exceptions::ErrorCode INVALID_PRODUCT_TYPE = 1270;
        const exceptions::ErrorCode INVALID_ORDER_STATE = 1280;
        const exceptions::ErrorCode INVALID_ORDER_SOURCE = 1290;
        const exceptions::ErrorCode
INVALID_BRANCH_SEQUENCE_NUMBER = 1300;
        const exceptions::ErrorCode MISSING_LISTENER = 1310;
        const exceptions::ErrorCode BUSINESS_DAY_NOT_STARTED =
1320;
        const exceptions::ErrorCode INVALID_FIELD_LENGTH = 1330;
        const exceptions::ErrorCode INVALID_STRATEGY_LEG = 1340;
```

```
        const exceptions::ErrorCode DUPLICATE_STRATEGY_LEG =
1350;
        const exceptions::ErrorCode INVALID_LEG_CONTINGENCY =
1360;
        const exceptions::ErrorCode INVALID_CANCEL_REQUEST =
1370;
        const exceptions::ErrorCode INVALID_VERSION = 1380;
        const exceptions::ErrorCode INVALID_LOGIN_MODE = 1390;
        const exceptions::ErrorCode
GMD_LISTENER_ALREADY_REGISTERED = 1400;
        const exceptions::ErrorCode INVALID_TRADE_SOURCE = 1410;
        const exceptions::ErrorCode INVALID_TRADE_TYPE = 1420;
        const exceptions::ErrorCode NO_REMAINING_QUANTITY = 1430;
    };

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

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

    interface TransactionFailedCodes  {
        const exceptions::ErrorCode CREATE_FAILED = 3000;
        const exceptions::ErrorCode UPDATE_FAILED = 3010;
        const exceptions::ErrorCode ACTION_VETOED = 3020;
        const exceptions::ErrorCode INSUFFICIENT_QUANTITY = 3030;
        const exceptions::ErrorCode REMAINING_QUANTITY_MISMATCH =
3040;

         const exceptions::ErrorCode INVALID_STATE_CHANGE = 3050;
        const exceptions::ErrorCode INCOMPLETE_STATE_CHANGE =
3060;
        const exceptions::ErrorCode NO_REMAINING_QUANTITY = 3070
    };

    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;
    };
```

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

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

Summary of Changes to CMi IDL Beta 2

This section describes the changes in the IDL between the CMi Beta 2 release and CMi Beta 1 release.

cmi.idl

MarketQuery

No change.

Quote

No change.

OrderQuery

No change.

ProductQuery

No change.

ProductDefinition

No change.

OrderEntry

No change.

UserPreferenceQuery

No change.

Administrator

No change.

TradingSession

No change.

UserHistory

No change.

UserTradingParameters

No change.

UserSessionManager

No change.

UserAccess

No change.

cmiCallBack.idl

No change.

cmiAdmin.idl

No change.

cmiMarketData.idl

No change.

cmiOrder.idl

Added a new typedef for NBBOProtectionType.

Added a new field in the orderEntryStruct: orderNBBOProtectionType.

cmiProduct.idl

No change.

cmiQuote.idl

No change.

cmiSession.idl

No change.

cmiStrategy.idl

No change.

cmiTraderActivity.idl

No change.

cmiUser.idl

Added a new typedef for exchange.

cmiUtil.idl

No change.

cmiConstants.idl

Added a constant for Stop Limit order.

Added CTI origin types.

Added the EXCHANGE_BROKER role.

Added an interface source for COMPASS.

Added values for orderNBBOProtectionType.

Added a new ExchangeStrings interface.

Changes in the Beta 2 IDL reflect the new OneChicago Exchange constant as "ONE". This documentation release references the OneChicago Exchange constant as "OCX" instead of "ONE". Updated documentation to reflect the new constant change will be available in the production version of the CMi, scheduled for release in April 2002.

cmiErrorCodes.idl

Added a new error code: LOST_CONNECTION

IDL Changes for Beta 2

IDL Changes are shown as follows:

Code that is unchanged from the previous version is shown in regular text:

```
interface MarketQuery
    {
        void subscribeRecapForClass(
            in cmiSession::TradingSessionName sessionName,
            in cmiProduct::ClassKey classKey,
            in cmiCallback::CMIRecapConsumer clientListener)
                raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::DataValidationException
                );
```

IDL that has been added is shown in red inside a text box:

```
void aNewMethodName(
            in cmiSession::TradingSessionName sessionName,
            in cmiProduct::ProductKey productKey,
            in cmiCallback::CMIOrderBookConsumer
orderBookConsumer)
                raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::DataValidationException,
                    exceptions::NotFoundException
            );
```

IDL that has been deleted is shown in grey in strike through inside a text box:

```
void someDeletedOperation(
            in cmiSession::TradingSessionName sessionName,
            in cmiProduct::ProductKey productKey,
            in cmiCallback::CMIOrderBookConsumer
orderBookConsumer)
                raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::DataValidationException,
                    exceptions::NotFoundException
```

cmi.idl

MarketQuery Interface

No change between Beta 1 and Beta 2.

Quote Interface

No change between Beta 1 and Beta 2.

OrderQuery Interface

No change between Beta 1 and Beta 2.

ProductQuery Interface

No change between Beta 1 and Beta 2.

ProductDefinition Interface

No change between Beta 1 and Beta 2.

OrderEntry Interface

No change between Beta 1 and Beta 2.

UserPreferenceQuery Interface

No change between Beta 1 and Beta 2.

Administrator Interface

No change between Beta 1 and Beta 2.

TradingSession Interface

No change between Beta 1 and Beta 2.

UserHistory Interface

No change between Beta 1 and Beta 2.

UserTradingParameters Interface

No change between Beta 1 and Beta 2.

UserSessionManager Interface

No change between Beta 1 and Beta 2.

UserAccess Interface

No change between Beta 1 and Beta 2.

cmiCallback.idl

No change between Beta 1 and Beta 2.

cmiAdmin.idl

No change between Beta 1 and Beta 2.

cmiMarketData.idl

No change between Beta 1 and Beta 2.

cmiOrder.idl

```
module cmiOrder
{
    typedef short ContingencyType;
    typedef short OrderState;
    typedef char  TimeInForce;
    typedef char  PositionEffect;
    typedef char  OriginType;
    typedef char  Coverage;
    typedef boolean  CrossingIndicator;
    typedef short CancelType;
    typedef short NBBOProtectionType;

    struct OrderContingencyStruct
    {
        cmiOrder::ContingencyType type;
        cmiUtil::PriceStruct price;
        long volume;
    };

    struct OrderIdStruct
    {
        cmiUser::ExchangeFirmStruct executingOrGiveUpFirm;
        string branch;
        long branchSequenceNumber;
        string correspondentFirm;
        string orderDate; // YYYYMMDD format
        long highCboeId;
        long lowCboeId;
    };

    struct OrderEntryStruct
    {
        cmiUser::ExchangeFirmStruct executingOrGiveUpFirm;
        string branch;
```

```
            long branchSequenceNumber;
            string correspondentFirm;
            string orderDate; // YYYYMMDD format

            cmiUser::ExchangeAcronymStruct originator;
            long originalQuantity;
            cmiProduct::ProductKey productKey;
            cmiUtil::Side side;
            cmiUtil::PriceStruct price;
            cmiOrder::TimeInForce timeInForce;
            cmiUtil::DateTimeStruct expireTime;
            cmiOrder::OrderContingencyStruct contingency;
            cmiUser::ExchangeFirmStruct cmta;
            string extensions;
            string account;
            string subaccount;
            cmiOrder::PositionEffect positionEffect;
            cmiOrder::CrossingIndicator cross;
            cmiOrder::OriginType orderOriginType;
            cmiOrder::Coverage coverage;
            cmiOrder::NBBOProtectionType orderNBBOProtectionType;
            string optionalData;
            string userAssignedId;
            cmiSession::TradingSessionNameSequence sessionNames;
        };
        typedef sequence <OrderEntryStruct> OrderEntryStructSequence;


        struct LegOrderEntryStruct
        {
            cmiProduct::ProductKey productKey;
            cmiUtil::PriceStruct mustUsePrice;
            cmiUser::ExchangeFirmStruct clearingFirm;
            cmiOrder::Coverage coverage;
            cmiOrder::PositionEffect positionEffect;
        };
        typedef sequence <LegOrderEntryStruct>
    LegOrderEntryStructSequence;


        struct LegOrderDetailStruct
        {
            cmiProduct::ProductKey productKey;
            cmiUtil::PriceStruct mustUsePrice;
            cmiUser::ExchangeFirmStruct clearingFirm;
            cmiOrder::Coverage coverage;
            cmiOrder::PositionEffect positionEffect;
            cmiUtil::Side side;
            long originalQuantity;
            long tradedQuantity;
            long cancelledQuantity;
            long leavesQuantity;
        };
        typedef sequence <LegOrderDetailStruct>
    LegOrderDetailStructSequence;
```

```
struct OrderStruct
{
    OrderIdStruct orderId;
    cmiUser::ExchangeAcronymStruct originator;

    // Fields from the OrderEntryStruct
    long originalQuantity;
    cmiProduct::ProductKey productKey;
    cmiUtil::Side side;
    cmiUtil::PriceStruct price;
    cmiOrder::TimeInForce timeInForce;
    cmiUtil::DateTimeStruct expireTime;
    cmiOrder::OrderContingencyStruct contingency;
    cmiUser::ExchangeFirmStruct cmta;
    string extensions;
    string account;
    string subaccount;
    cmiOrder::PositionEffect positionEffect;
    cmiOrder::CrossingIndicator cross;
    cmiOrder::OriginType orderOriginType;
    cmiOrder::Coverage coverage;
    cmiOrder::NBBOProtectionType orderNBBOProtectionType;
    string optionalData;

    // Additional Order Fields
    string userId;
    cmiUser::ExchangeAcronymStruct userAcronym;
    cmiProduct::ProductType productType;
    cmiProduct::ClassKey classKey;
    cmiUtil::DateTimeStruct receivedTime;
    cmiOrder::OrderState state;
    long tradedQuantity;
    long cancelledQuantity;

    long leavesQuantity;
    cmiUtil::PriceStruct averagePrice;
    long sessionTradedQuantity;
    long sessionCancelledQuantity;
    cmiUtil::PriceStruct sessionAveragePrice;

    string orsId;
    cmiUtil::Source source;
    cmiOrder::OrderIdStruct crossedOrder;
    long transactionSequenceNumber;
    string userAssignedId;
    cmiSession::TradingSessionNameSequence sessionNames;
    cmiSession::TradingSessionName activeSession;
    cmiOrder::LegOrderDetailStructSequence legOrderDetails;
};
typedef sequence <OrderStruct> OrderStructSequence;

struct OrderDetailStruct
{
```

```
            cmiProduct::ProductNameStruct productInformation;
            cmiUtil::UpdateStatusReason statusChange;
            cmiOrder::OrderStruct orderStruct;
        };
        typedef sequence <OrderDetailStruct>
    OrderDetailStructSequence;

        struct CancelReportStruct
        {
            cmiOrder::OrderIdStruct orderId;
            cmiUtil::ReportType cancelReportType;
            cmiProduct::ProductKey productKey;
            cmiSession::TradingSessionName sessionName;
            long cancelledQuantity;
            long tlcQuantity;
            long mismatchedQuantity;
            cmiUtil::DateTimeStruct timeSent;
            string orsId;
            long totalCancelledQuantity;
            boolean nothingDone;
            long transactionSequenceNumber;
            string userAssignedCancelId;
        };
        typedef sequence <CancelReportStruct>
    CancelReportStructSequence;

        struct CancelRequestStruct
        {
            cmiOrder::OrderIdStruct orderId;
            cmiSession::TradingSessionName sessionName;
            string userAssignedCancelId;
            cmiOrder::CancelType cancelType;
            long quantity;
        };

        struct ContraPartyStruct
        {
            cmiUser::ExchangeAcronymStruct user;
            cmiUser::ExchangeFirmStruct firm;
            long quantity;
        };
        typedef sequence <ContraPartyStruct>
    ContraPartyStructSequence;

        struct FilledReportStruct
        {
            cmiUtil::CboeIdStruct tradeId;
            cmiUtil::ReportType fillReportType;
            cmiUser::ExchangeFirmStruct executingOrGiveUpFirm;
            string userId;
            cmiUser::ExchangeAcronymStruct  userAcronym;
            cmiProduct::ProductKey productKey;
            cmiSession::TradingSessionName sessionName;
            long tradedQuantity;
            long leavesQuantity;
```

```
            cmiUtil::PriceStruct price;
            cmiUtil::Side side;
            string orsId;
            string executingBroker;
            cmiUser::ExchangeFirmStruct cmta;
            string account;
            string subaccount;
            cmiUser::ExchangeAcronymStruct originator;
            string optionalData;
            string userAssignedId;
            string extensions;
            cmiOrder::ContraPartyStructSequence contraParties;
            cmiUtil::DateTimeStruct timeSent;
            cmiOrder::PositionEffect positionEffect;
            long transactionSequenceNumber;
        };
        typedef sequence <FilledReportStruct>
    FilledReportStructSequence;

        struct OrderFilledReportStruct
        {
            cmiOrder::OrderDetailStruct filledOrder;
            cmiOrder::FilledReportStructSequence filledReport;
        };
        typedef sequence <OrderFilledReportStruct>
    OrderFilledReportStructSequence;

        struct OrderCancelReportStruct
        {
            cmiOrder::OrderDetailStruct cancelledOrder;
            cmiOrder::CancelReportStructSequence cancelReport;
        };
        typedef sequence <OrderCancelReportStruct>
    OrderCancelReportStructSequence;

        struct PendingOrderStruct {
            cmiProduct::PendingNameStruct pendingProductName;
            cmiOrder::OrderStruct pendingOrder;
            cmiOrder::OrderStruct currentOrder;
        };
        typedef sequence <PendingOrderStruct>
    PendingOrderStructSequence;

        struct BustReportStruct
        {
            cmiUtil::CboeIdStruct tradeId;
            cmiUtil::ReportType bustReportType;
            cmiSession::TradingSessionName sessionName;
            cmiUser::ExchangeFirmStruct executingOrGiveUpFirm;
            string userId;
            cmiUser::ExchangeAcronymStruct  userAcronym;
            long bustedQuantity;
            cmiUtil::PriceStruct price;
            cmiProduct::ProductKey productKey;
            cmiUtil::Side side;
```

```
            cmiUtil::DateTimeStruct timeSent;
            long reinstateRequestedQuantity;
            long transactionSequenceNumber;
        };
        typedef sequence <BustReportStruct> BustReportStructSequence;

        struct OrderBustReportStruct
        {
            cmiOrder::OrderDetailStruct bustedOrder;
            cmiOrder::BustReportStructSequence bustedReport;
        };
        typedef sequence <OrderBustReportStruct>
    OrderBustReportStructSequence;

        struct BustReinstateReportStruct
        {
            cmiUtil::CboeIdStruct tradeId;
            long bustedQuantity;
            long reinstatedQuantity;
            long totalRemainingQuantity;
            cmiUtil::PriceStruct price;
            cmiProduct::ProductKey productKey;
            cmiSession::TradingSessionName sessionName;
            cmiUtil::Side side;
            cmiUtil::DateTimeStruct timeSent;
            long transactionSequenceNumber;
        };

        typedef sequence <BustReinstateReportStruct>
    BustReinstateReportStructSequence;

        struct OrderBustReinstateReportStruct
        {
            cmiOrder::OrderDetailStruct reinstatedOrder;
            cmiOrder::BustReinstateReportStruct bustReinstatedReport;
        };

        typedef sequence <OrderBustReinstateReportStruct>
    OrderBustReinstateReportStructSequence;
    };
```

cmiProduct.idl

No change between Beta 1 and Beta 2.

cmiQuote.idl

No change between Beta 1 and Beta 2.

cmiSession.idl

No change between Beta 1 and Beta 2.

cmiStrategy.idl

No change between Beta 1 and Beta 2.

cmiTraderActivity.idl

No change between Beta 1 and Beta 2.

cmiUser.idl

```
module cmiUser
{

    typedef char UserRole;
    typedef char LoginSessionMode;
    typedef string Exchange;

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

    struct ExchangeAcronymStruct
    {
string exchange;
   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 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;
        };

    };
```

cmiUtil.idl

No change between Beta 1 and Beta 2.

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 LoginSessionTypes
    {
        const cmiSession::LoginSessionType PRIMARY = 1;
        const cmiSession::LoginSessionType SECONDARY = 2;
    };

    interface MarketDataHistoryEntryTypes
    {
        const cmiMarketData::MarketDataHistoryEntryType
QUOTE_ENTRY = 1;
        const cmiMarketData::MarketDataHistoryEntryType
PRICE_REPORT_ENTRY = 2;
        const cmiMarketData::MarketDataHistoryEntryType
EXPECTED_OPEN_PRICE = 3;
        const cmiMarketData::MarketDataHistoryEntryType
MARKET_CONDITION_ENTRY = 4;
        const cmiMarketData::MarketDataHistoryEntryType
UNSIZED_QUOTE_ENTRY = 5;
    };

    interface MarketChangeReasons
    {
        const cmiMarketData::MarketChangeReason EXCHANGE = 1;
        const cmiMarketData::MarketChangeReason NBBO = 2;
        const cmiMarketData::MarketChangeReason COMBINED = 3;
    };

    interface QueryDirections
    {
        const cmiUtil::QueryDirection QUERY_FORWARD = 1;
        const cmiUtil::QueryDirection QUERY_BACKWARD = 2;
    };


    interface PriceDisplayTypes
    {
        const cmiProduct::PriceDisplayType FRACTION = 1;
        const cmiProduct::PriceDisplayType DECIMAL = 2;
    };

    interface OptionTypes
    {
        const cmiProduct::OptionType CALL = 'C';
        const cmiProduct::OptionType PUT = 'P';
    };

    interface ProductTypes
    {
        const cmiProduct::ProductType COMMODITY = 1;
        const cmiProduct::ProductType DEBT = 2;
        const cmiProduct::ProductType EQUITY = 3;
        const cmiProduct::ProductType FUTURE = 4;
        const cmiProduct::ProductType INDEX = 5;
        const cmiProduct::ProductType LINKED_NOTE = 6;
```

```
        const cmiProduct::ProductType OPTION = 7;
        const cmiProduct::ProductType UNIT_INVESTMENT_TRUST = 8;
        const cmiProduct::ProductType VOLATILITY_INDEX = 9;
        const cmiProduct::ProductType WARRANT = 10;
        const cmiProduct::ProductType STRATEGY = 11;
    };

    interface ProductStates
    {
        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;
    };

    interface ListingStates
    {
        const cmiProduct::ListingState ACTIVE = 1;
        const cmiProduct::ListingState INACTIVE = 2;
        const cmiProduct::ListingState UNLISTED = 3;
        const cmiProduct::ListingState OBSOLETE = 4;
    };

    interface ClassStates
    {
        const cmiSession::ClassState NOT_IMPLEMENTED = 1;
    };

    interface TradingSessionStates
    {
        const cmiSession::TradingSessionState CLOSED = 1;
        const cmiSession::TradingSessionState OPEN = 2;
    };

    interface TradingSessionType
    {
        const cmiSession::TradingSessionState DAY = 1;
        const cmiSession::TradingSessionState EVENING = 2;
    };

    interface TradingSessionMethod
    {
        const cmiSession::TradingSessionState SBT = 1;
        const cmiSession::TradingSessionState OPENOUTCRY = 2;
    };

    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 REMOVED = 11;
            const cmiUtil::UpdateStatusReason QRM_REMOVED = 12;
        };

        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
        };

        interface VolumeTypes
        {
            const cmiMarketData::VolumeType LIMIT = 1; // Limit (no
contingency)
            const cmiMarketData::VolumeType AON = 2; // All or None
            const cmiMarketData::VolumeType FOK = 3; // Fill or Kill
            const cmiMarketData::VolumeType IOC = 4; // Immediate or
Cancel
        };

        interface OrderStates
        {
            const cmiOrder::OrderState BOOKED = 1;
            const cmiOrder::OrderState CANCEL = 2;
            const cmiOrder::OrderState FILL = 3;
            const cmiOrder::OrderState OPEN_OUTCRY = 4;
            const cmiOrder::OrderState INACTIVE = 5;
            const cmiOrder::OrderState ACTIVE = 6;
            const cmiOrder::OrderState EXPIRED = 7;
            const cmiOrder::OrderState PURGED = 8;
            const cmiOrder::OrderState REMOVED = 9;
```

```
                    };

                    interface RFQTypes
                    {
                        const cmiQuote::RFQType MANUAL = 1;
                        const cmiQuote::RFQType SYSTEM = 2;
                    };

                    interface Sides
                    {
                        const cmiUtil::Side BUY  = 'B';
                        const cmiUtil::Side SELL = 'S';
                        const cmiUtil::Side BID  = 'B';
                        const cmiUtil::Side ASK  = 'A';
                        const cmiUtil::Side AS_DEFINED = 'D';
                        const cmiUtil::Side OPPOSITE = 'O';
                    };

                    interface TimesInForce
                    {
                        const cmiOrder::TimeInForce GTC = 'G';
                        const cmiOrder::TimeInForce DAY = 'D';
                        const cmiOrder::TimeInForce GTD = 'T'; // Good until
                datetime
};


                    interface PositionEffects
                    {
                        const cmiOrder::PositionEffect OPEN = 'O';
                        const cmiOrder::PositionEffect CLOSED = 'C';
                        const cmiOrder::PositionEffect NOTAPPLICABLE = 'N';
                    };

                    interface OrderOrigins
                    {
                     const cmiOrder::OriginType CUSTOMER = 'C';
                    //CTI Equivalent - Non Member, Customer Segregated Account
                     const cmiOrder::OriginType FIRM = 'F';
                    //CTI Equivalent - Firm Trader, House Account
                     const cmiOrder::OriginType BROKER_DEALER = 'B';
                     const cmiOrder::OriginType CUSTOMER_BROKER_DEALER = 'X';
                     const cmiOrder::OriginType MARKET_MAKER = 'M';
                     const cmiOrder::OriginType CTI1Origin1 = 'V';
                    //Member, Customer Segregated Account
                     const cmiOrder::OriginType CTI1Origin2 = 'E';
                    //Member, House Account
                     const cmiOrder::OriginType CTI1Origin5 = 'Q';
                    //Member, SIPC Protected Account
                     const cmiOrder::OriginType CTI3Origin1 = 'G';
                    //User Proxy for trader, Customer Segregated Account
                     const cmiOrder::OriginType CTI3Origin2 = 'H';
                    //User Proxy for trader, House Account
                     const cmiOrder::OriginType CTI3Origin5 = 'R';
                    //User Proxy for trader, SIPC Protected Account
```

```
        const cmiOrder::OriginType CTI4Origin2 = 'O';
       //Non Member, House Account
        const cmiOrder::OriginType CTI4Origin5 = 'T';
       //Non Member, SIPC Protected Account
};

    interface UserRoles
    {
        const cmiUser::UserRole FIRM = 'F';
        const cmiUser::UserRole BROKER_DEALER = 'B';
        const cmiUser::UserRole CUSTOMER_BROKER_DEALER = 'X';
        const cmiUser::UserRole MARKET_MAKER = 'M';
        const cmiUser::UserRole HELP_DESK = 'H';
        const cmiUser::UserRole DPM_ROLE = 'D';
        const cmiUser::UserRole UNKNOWN_ROLE = 'K';
        const cmiUser::UserRole CLASS_DISPLAY = 'C';
        const cmiUser::UserRole FIRM_DISPLAY = 'R';
        const cmiUser::UserRole EXCHANGE_BROKER = 'E';
    };

    interface EntryTypes
    {
        const cmiUtil::EntryType ADD = 'A';
        const cmiUtil::EntryType CANCEL = 'C';
        const cmiUtil::EntryType CANCEL_REPLACE = 'R';
        const cmiUtil::EntryType FILL = 'F';
        const cmiUtil::EntryType BOOK = 'B';
        const cmiUtil::EntryType STATE = 'S';
        const cmiUtil::EntryType PRICE_ADJUST = 'P';
        const cmiUtil::EntryType UPDATE = 'U';
        const cmiUtil::EntryType BUST = 'K';
    };

    interface CoverageTypes
    {
        const cmiOrder::Coverage UNSPECIFIED = 'B';
        const cmiOrder::Coverage COVERED = 'C';
        const cmiOrder::Coverage UNCOVERED = 'U';
    };

    interface Sources
    {
        const cmiUtil::Source TPF = 'T';
        const cmiUtil::Source SBT = 'S';
        const cmiUtil::Source COMPASS = 'C';
    };

    interface PriceTypes
    {
        const cmiUtil::PriceType NO_PRICE = 1;
        const cmiUtil::PriceType LIMIT = 2;
        const cmiUtil::PriceType VALUED = 2;
        const cmiUtil::PriceType MARKET = 3;
    };
```

```
interface ExpirationStyles
{
    const cmiProduct::ExpirationStyle EUROPEAN = 'E';
    const cmiProduct::ExpirationStyle AMERICAN = 'A';
};
```

```
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
    };
```

```
 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;
    };

    interface PriceAdjustmentTypes
    {
        const cmiProduct::PriceAdjustmentType SPLIT = 1;
        const cmiProduct::PriceAdjustmentType DIVIDEND_CASH = 2;
        const cmiProduct::PriceAdjustmentType DIVIDEND_PERCENT =
3;
        const cmiProduct::PriceAdjustmentType DIVIDEND_STOCK = 4;
        const cmiProduct::PriceAdjustmentType LEAP_ROLLOVER = 5;
        const cmiProduct::PriceAdjustmentType MERGER = 6;
        const cmiProduct::PriceAdjustmentType SYMBOL_CHANGE = 7;
        const cmiProduct::PriceAdjustmentType COMMON_DISTRIBUTION
= 8;
    };


    interface PriceAdjustmentActions
    {
        const cmiProduct::PriceAdjustmentAction
PRICE_ADJUSTMENT_UPDATE = 1;
        const cmiProduct::PriceAdjustmentAction
PRICE_ADJUSTMENT_DELETE = 2;
        const cmiProduct::PriceAdjustmentAction
PRICE_ADJUSTMENT_CREATE = 3;
        const cmiProduct::PriceAdjustmentAction
PRICE_ADJUSTMENT_MOVE = 4;
    };

    interface PriceScale
    {
        const long DEFAULT_SCALE = 1000000000;
    };

    interface ProductClass
    {
        const long DEFAULT_CLASS_KEY = 0;
    };

    interface OrderCancelTypes
    {
        const cmiOrder::CancelType DESIRED_CANCEL_QUANTITY    =
1;
        const cmiOrder::CancelType DESIRED_REMAINING_QUANTITY =
2;
        const cmiOrder::CancelType CANCEL_ALL_QUANTITY        =
3;
    };


    interface ActivityTypes
    {
        // Order Activity Events
```

```
        const cmiTraderActivity::ActivityType  NEW_ORDER
= 1;
        const cmiTraderActivity::ActivityType  FILL_ORDER
= 2;
        const cmiTraderActivity::ActivityType  CANCEL_ORDER
= 3;
        const cmiTraderActivity::ActivityType  BUST_ORDER_FILL
= 4;
        const cmiTraderActivity::ActivityType
BUST_REINSTATE_ORDER    = 5;
        const cmiTraderActivity::ActivityType
CANCEL_REPLACE_ORDER    = 6;
        const cmiTraderActivity::ActivityType  UPDATE_ORDER
= 7;
        const cmiTraderActivity::ActivityType  BOOK_ORDER
= 8;
        const cmiTraderActivity::ActivityType  STATE_CHANGE_ORDER
= 9;
        const cmiTraderActivity::ActivityType  PRICE_ADJUST_ORDER
= 10;
        const cmiTraderActivity::ActivityType  CANCEL_ALL_ORDERS
= 11;

        // Strategy Order leg activity types
        const cmiTraderActivity::ActivityType
NEW_ORDER_STRATEGY_LEG      = 51;
        const cmiTraderActivity::ActivityType  FILL_STRATEGY_LEG
= 52;
        const cmiTraderActivity::ActivityType
CANCEL_STRATEGY_LEG         = 53;
        const cmiTraderActivity::ActivityType
BUST_STRATEGY_LEG_FILL      = 54;
        const cmiTraderActivity::ActivityType
BUST_REINSTATE_STRATEGY_LEG = 55;
        const cmiTraderActivity::ActivityType
UPDATE_STRATEGY_LEG         = 57;
        const cmiTraderActivity::ActivityType
PRICE_ADJUST_ORDER_LEG      = 60;

        // Quote Activity Events
        const cmiTraderActivity::ActivityType  NEW_QUOTE
= 101;
        const cmiTraderActivity::ActivityType  FILL_QUOTE
= 102;
        const cmiTraderActivity::ActivityType  CANCEL_QUOTE
= 103;
        const cmiTraderActivity::ActivityType  CANCEL_ALL_QUOTES
= 104;
        const cmiTraderActivity::ActivityType
SYSTEM_CANCEL_QUOTE     = 105;
        const cmiTraderActivity::ActivityType  UPDATE_QUOTE
= 106;
        const cmiTraderActivity::ActivityType  BUST_QUOTE_FILL
= 107;
```

```
        // Strategy Quote leg activity types
        const cmiTraderActivity::ActivityType  QUOTE_LEG_FILL
= 152;
        const cmiTraderActivity::ActivityType
BUST_QUOTE_LEG_FILL     = 157;

        // RFQ Activity Events
        const cmiTraderActivity::ActivityType  NEW_RFQ
= 201;

    };

    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;
    };

    interface ActivityReasons
    {
        const cmiTraderActivity::ActivityReason
NOTHING_DONE               = 1; // used with CANCEL_REASON
        const cmiTraderActivity::ActivityReason         USER
= 2; // used with CANCEL_REASON
        const cmiTraderActivity::ActivityReason         SYSTEM
= 3; // used with CANCEL_REASON
    };

    interface StrategyTypes
    {
        const cmiStrategy::StrategyType UNKNOWN = 1;

        const cmiStrategy::StrategyType STRADDLE = 2;
        const cmiStrategy::StrategyType PSEUDO_STRADDLE = 3;
        const cmiStrategy::StrategyType VERTICAL = 4;
        const cmiStrategy::StrategyType RATIO = 5;
        const cmiStrategy::StrategyType TIME = 6;
        const cmiStrategy::StrategyType DIAGONAL = 7;
        const cmiStrategy::StrategyType COMBO = 8;
    };
```

```
        interface ReportTypes
        {
            const cmiUtil::ReportType REGULAR_REPORT = 1;
            const cmiUtil::ReportType STRATEGY_REPORT = 2;
            const cmiUtil::ReportType STRATEGY_LEG_REPORT = 3;
        };

        interface BookDepthTypes
        {
            const cmiMarketData::BookDepthUpdateType DELETE_PRICE =
'D';        // delete book price
            const cmiMarketData::BookDepthUpdateType INSERT_PRICE =
'I';        // new book price
            const cmiMarketData::BookDepthUpdateType UPDATE_PRICE =
'U';        // book price update
        };

        interface OrderNBBOProtectionTypes
        {
            const cmiOrder::NBBOProtectionType NONE = 1;
            const cmiOrder::NBBOProtectionType FULL = 2;
    };
};
```

cmiErrorCodes.idl

```
module cmiErrorCodes
{
    interface DataValidationCodes  {
        const exceptions::ErrorCode DUPLICATE_ID = 1000;
        const exceptions::ErrorCode INVALID_VOLUME = 1010;
        const exceptions::ErrorCode INVALID_TIME = 1020;
        const exceptions::ErrorCode INCOMPLETE_QUOTE = 1030;
        const exceptions::ErrorCode INVALID_QUANTITY = 1040;
        const exceptions::ErrorCode INVALID_STRATEGY = 1060;
        const exceptions::ErrorCode INVALID_SPREAD = 1070;
        const exceptions::ErrorCode INVALID_MEMBERKEY = 1080;
        const exceptions::ErrorCode INVALID_PRODUCT = 1090;
        const exceptions::ErrorCode INVALID_SESSION = 1100;
        const exceptions::ErrorCode INVALID_STATE = 1110;
        const exceptions::ErrorCode PREFERENCE_PATH_MISMATCH =
1120;
        const exceptions::ErrorCode INVALID_ORDER_ID = 1130;
        const exceptions::ErrorCode LISTENER_ALREADY_REGISTERED =
1140;
        const exceptions::ErrorCode INVALID_SIDE = 1150;
        const exceptions::ErrorCode INVALID_PRICE = 1160;
        const exceptions::ErrorCode INVALID_UPDATE_ATTEMPT =
1170;
        const exceptions::ErrorCode INVALID_ORIGINATOR = 1180;
        const exceptions::ErrorCode INVALID_ACCOUNT = 1200;
        const exceptions::ErrorCode INVALID_EXECUTING_GIVEUP_FIRM
= 1210;
```

```
        const exceptions::ErrorCode INVALID_CONTINGENCY_TYPE =
1220;
        const exceptions::ErrorCode INVALID_TIME_IN_FORCE = 1230;
        const exceptions::ErrorCode INVALID_POSITION_EFFECT =
1240;
        const exceptions::ErrorCode INVALID_ORIGIN_TYPE = 1250;
      const exceptions::ErrorCode INVALID_COVERAGE = 1260;
      const exceptions::ErrorCode INVALID_PRODUCT_TYPE = 1270;
      const exceptions::ErrorCode INVALID_ORDER_STATE = 1280;
      const exceptions::ErrorCode INVALID_ORDER_SOURCE = 1290;
      const exceptions::ErrorCode INVALID_BRANCH_SEQUENCE_NUMBER = 1300;
      const exceptions::ErrorCode MISSING_LISTENER = 1310;
      const exceptions::ErrorCode BUSINESS_DAY_NOT_STARTED = 1320;
      const exceptions::ErrorCode INVALID_FIELD_LENGTH = 1330;
      const exceptions::ErrorCode INVALID_STRATEGY_LEG = 1340;
      const exceptions::ErrorCode DUPLICATE_STRATEGY_LEG = 1350;
      const exceptions::ErrorCode INVALID_LEG_CONTINGENCY = 1360;
      const exceptions::ErrorCode INVALID_CANCEL_REQUEST = 1370;
      const exceptions::ErrorCode INVALID_VERSION = 1380;
      const exceptions::ErrorCode INVALID_LOGIN_MODE = 1390;
      const exceptions::ErrorCode GMD_LISTENER_ALREADY_REGISTERED = 1400;
  };

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

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

   interface TransactionFailedCodes  {
       const exceptions::ErrorCode CREATE_FAILED = 3000;
       const exceptions::ErrorCode UPDATE_FAILED = 3010;
       const exceptions::ErrorCode ACTION_VETOED = 3020;
       const exceptions::ErrorCode INSUFFICIENT_QUANTITY = 3030;
       const exceptions::ErrorCode REMAINING_QUANTITY_MISMATCH =
3040;
       const exceptions::ErrorCode INVALID_STATE_CHANGE = 3050;
       const exceptions::ErrorCode INCOMPLETE_STATE_CHANGE =
3060;
       const exceptions::ErrorCode NO_REMAINING_QUANTITY = 3070;
   };

   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;
   };

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

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

Summary of Changes to CMi IDL Beta 1

This section describes the changes in the idl between the CMi Beta 1 release and CMi Documentation Only release.

cmi.idl

MarketQuery

Added a NotAcceptedException to the getBookDepth method

Quote

No change.

OrderQuery

No change.

ProductQuery

No change.

ProductDefinition

No change.

OrderEntry

Added new attributes in the cancel replace struct

Removed attributes in the order ID struct

UserPreferenceQuery

No change.

Administrator

No change.

TradingSession

No change.

UserHistory

No change.

UserTradingParameters

No change.

UserSessionManager

No change.

UserAccess

No change.

cmiCallBack.idl

No change.

cmiAdmin.idl

No change.

cmiMarketData.idl

Removed the Order Book Side Struct.

cmiOrder.idl

No change.

cmiProduct.idl

No change.

cmiQuote.idl

No change.

cmiSession.idl

No change.

cmiStrategy.idl

No change.

cmiTraderActivity.idl

No change.

cmiUser.idl

No change.

cmiUtil.idl

No change.

cmiConstants.idl

No change.

cmiErrorCodes.idl

Added new attributes for DataValidation, TransactionFailed, and NotAcceptedCodes.

IDL Changes for Beta 1

IDL Changes are shown as follows:

Code that is unchanged from the previous version is shown in regular text:

```
interface MarketQuery
    {
        void subscribeRecapForClass(
            in cmiSession::TradingSessionName sessionName,
            in cmiProduct::ClassKey classKey,
            in cmiCallback::CMIRecapConsumer clientListener)
                raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::DataValidationException
                );
```

IDL that has been added is shown in red inside a text box:

```
void aNewMethodName(
            in cmiSession::TradingSessionName sessionName,
            in cmiProduct::ProductKey productKey,
            in cmiCallback::CMIOrderBookConsumer
orderBookConsumer)
                raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::DataValidationException,
                    exceptions::NotFoundException
            );
```

IDL that has been deleted is shown in grey in strike through inside a text box:

```
void someDeletedOperation(
            in cmiSession::TradingSessionName sessionName,
            in cmiProduct::ProductKey productKey,
            in cmiCallback::CMIOrderBookConsumer
orderBookConsumer)
                raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::DataValidationException,
                    exceptions::NotFoundException
            );
```

cmi.idl

MarketQuery Interface

```
interface MarketQuery
    {
        void subscribeRecapForClass(
            in cmiSession::TradingSessionName sessionName,
            in cmiProduct::ClassKey classKey,
            in cmiCallback::CMIRecapConsumer clientListener)
                raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::DataValidationException
                );

        void subscribeRecapForProduct(
            in cmiSession::TradingSessionName sessionName,
            in cmiProduct::ProductKey productKey,
            in cmiCallback::CMIRecapConsumer clientListener)
                raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::DataValidationException
                );

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

        void subscribeCurrentMarketForProduct(
            in cmiSession::TradingSessionName sessionName,
            in cmiProduct::ProductKey productKey,
            in cmiCallback::CMICurrentMarketConsumer
clientListener)
                raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::DataValidationException
                );

         void subscribeNBBOForClass(
            in cmiSession::TradingSessionName sessionName,
```

```
                    in cmiProduct::ClassKey classKey,
                    in cmiCallback::CMINBBOConsumer clientListener)
                        raises(
                            exceptions::SystemException,
                            exceptions::CommunicationException,
                            exceptions::AuthorizationException,
                            exceptions::DataValidationException
                        );

                void subscribeNBBOForProduct(
                    in cmiSession::TradingSessionName sessionName,
                    in cmiProduct::ProductKey productKey,
                    in cmiCallback::CMINBBOConsumer clientListener)
                        raises(
                            exceptions::SystemException,
                            exceptions::CommunicationException,
                            exceptions::AuthorizationException,
                            exceptions::DataValidationException
                        );

                void unsubscribeCurrentMarketForProduct(
                    in cmiSession::TradingSessionName sessionName,
                    in cmiProduct::ProductKey productKey,
                    in cmiCallback::CMICurrentMarketConsumer
clientListener)
                        raises(
                            exceptions::SystemException,
                            exceptions::CommunicationException,
                            exceptions::AuthorizationException,
                            exceptions::DataValidationException
                        );

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

                 void unsubscribeNBBOForProduct(
                    in cmiSession::TradingSessionName sessionName,
                    in cmiProduct::ProductKey productKey,
                    in cmiCallback::CMINBBOConsumer clientListener)
                        raises(
                            exceptions::SystemException,
                            exceptions::CommunicationException,
                            exceptions::AuthorizationException,
                            exceptions::DataValidationException
                        );
```

```
void unsubscribeNBBOForClass(
    in cmiSession::TradingSessionName sessionName,
    in cmiProduct::ClassKey classKey,
    in cmiCallback::CMINBBOConsumer clientListener)
        raises(
            exceptions::SystemException,
            exceptions::CommunicationException,
            exceptions::AuthorizationException,
            exceptions::DataValidationException
        );

void subscribeTicker(
    in cmiSession::TradingSessionName sessionName,
    in cmiProduct::ProductKey productKey,
    in cmiCallback::CMITickerConsumer clientListener)
        raises(
            exceptions::SystemException,
            exceptions::CommunicationException,
            exceptions::AuthorizationException,
            exceptions::DataValidationException
        );

void unsubscribeRecapForProduct(
    in cmiSession::TradingSessionName sessionName,
    in cmiProduct::ProductKey productKey,
    in cmiCallback::CMIRecapConsumer clientListener)
        raises(
            exceptions::SystemException,
            exceptions::CommunicationException,
            exceptions::AuthorizationException,
            exceptions::DataValidationException
        );

void unsubscribeRecapForClass(
    in cmiSession::TradingSessionName sessionName,
    in cmiProduct::ClassKey classKey,
    in cmiCallback::CMIRecapConsumer clientListener)
        raises(
            exceptions::SystemException,
            exceptions::CommunicationException,
            exceptions::AuthorizationException,
            exceptions::DataValidationException
        );

void unsubscribeTicker(
    in cmiSession::TradingSessionName sessionName,
    in cmiProduct::ProductKey productKey,
    in cmiCallback::CMITickerConsumer clientListener)
        raises(
            exceptions::SystemException,
            exceptions::CommunicationException,
            exceptions::AuthorizationException,
            exceptions::DataValidationException
        );
```

```
        cmiMarketData::MarketDataHistoryStruct
getMarketDataHistoryByTime(
            in cmiSession::TradingSessionName sessionName,
            in cmiProduct::ProductKey productKey,
            in cmiUtil::DateTimeStruct startTime,
            in cmiUtil::QueryDirection direction)
                raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::DataValidationException,
                    exceptions::NotFoundException
                );

        void subscribeExpectedOpeningPrice(
            in cmiSession::TradingSessionName sessionName,
            in cmiProduct::ClassKey classKey,
            in cmiCallback::CMIExpectedOpeningPriceConsumer
clientListener)
                raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::DataValidationException
                );

        void unsubscribeExpectedOpeningPrice(
            in cmiSession::TradingSessionName sessionName,
            in cmiProduct::ClassKey classKey,
            in cmiCallback::CMIExpectedOpeningPriceConsumer
clientListener)
                raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::DataValidationException
                );

        cmiMarketData::BookDepthStruct getBookDepth(
            in cmiSession::TradingSessionName sessionName,
            in cmiProduct::ProductKey productKey)
                raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::DataValidationException,
                    exceptions::NotFoundException,
                    exceptions::NotAcceptedException
);

        void subscribeBookDepth(
            in cmiSession::TradingSessionName sessionName,
            in cmiProduct::ProductKey productKey,
```

```
                in cmiCallback::CMIOrderBookConsumer
            orderBookConsumer)
                    raises(
                        exceptions::SystemException,
                        exceptions::CommunicationException,
                        exceptions::AuthorizationException,
                        exceptions::DataValidationException,
                        exceptions::NotFoundException
                    );

                void unsubscribeBookDepth(
                    in cmiSession::TradingSessionName sessionName,
                    in cmiProduct::ProductKey productKey,
                    in cmiCallback::CMIOrderBookConsumer
            orderBookConsumer)
                    raises(
                        exceptions::SystemException,
                        exceptions::CommunicationException,
                        exceptions::AuthorizationException,
                        exceptions::DataValidationException,
                        exceptions::NotFoundException
                    );

                void subscribeBookDepthUpdate(
                    in cmiSession::TradingSessionName sessionName,
                    in cmiProduct::ProductKey productKey,
                    in cmiCallback::CMIOrderBookUpdateConsumer
            orderBookConsumer)
                    raises(
                        exceptions::SystemException,
                        exceptions::CommunicationException,
                        exceptions::AuthorizationException,
                        exceptions::DataValidationException,
                        exceptions::NotFoundException
                    );

                void unsubscribeBookDepthUpdate(
                    in cmiSession::TradingSessionName sessionName,
                    in cmiProduct::ProductKey productKey,
                    in cmiCallback::CMIOrderBookUpdateConsumer
            orderBookConsumer)
                    raises(
                        exceptions::SystemException,
                        exceptions::CommunicationException,
                        exceptions::AuthorizationException,
                        exceptions::DataValidationException,
                        exceptions::NotFoundException
                    );
            };
```

Quote Interface

No change between Version 2.0 Documentation Only and Version 2.0 Beta 1

OrderQuery Interface

No change between Version 2.0 Documentation Only and Version 2.0 Beta 1.

ProductQuery Interface

No change between Version 2.0 Documentation Only and Version 2.0 Beta 1.

ProductDefinition Interface

No change between Version 2.0 Documentation Only and Version 2.0 Beta 1.

OrderEntry Interface

```
interface OrderEntry {

        cmiOrder::OrderIdStruct acceptOrder(
            in cmiOrder::OrderEntryStruct anOrder)
                raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::DataValidationException,
                    exceptions::NotAcceptedException,
                    exceptions::TransactionFailedException,
                    exceptions::AlreadyExistsException
                );

        cmiOrder::OrderIdStruct acceptOrderByProductName(
            in cmiProduct::ProductNameStruct product,
            in cmiOrder::OrderEntryStruct anOrder)
                raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::DataValidationException,
                    exceptions::NotAcceptedException,
                    exceptions::TransactionFailedException,
                    exceptions::AlreadyExistsException
                );

        void acceptOrderCancelRequest(
            in cmiOrder::CancelRequestStruct cancelRequest)
                raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::DataValidationException,
                    exceptions::NotAcceptedException,
                    exceptions::TransactionFailedException
                );


        void acceptOrderUpdateRequest(
```

```
                in long currentRemainingQuantity,
                in cmiOrder::OrderEntryStruct updatedOrder)
                    raises(
                        exceptions::SystemException,
                        exceptions::CommunicationException,
                        exceptions::AuthorizationException,
                        exceptions::DataValidationException,
                        exceptions::NotAcceptedException,
                        exceptions::TransactionFailedException
                    );

            cmiOrder::OrderIdStruct acceptOrderCancelReplaceRequest(
                in cmiOrder::CancelRequestStruct cancelRequest,

in cmiOrder::OrderIdStruct orderId,
in long cancelQuantity,
             in cmiOrder::OrderEntryStruct newOrder)
                    raises(
                        exceptions::SystemException,
                        exceptions::CommunicationException,
                        exceptions::AuthorizationException,
                        exceptions::DataValidationException,
                        exceptions::NotAcceptedException,
                        exceptions::TransactionFailedException
                    );

            void acceptCrossingOrder(
                in cmiOrder::OrderEntryStruct buyCrossingOrder,
                in cmiOrder::OrderEntryStruct sellCrossingOrder)
                    raises(
                        exceptions::SystemException,
                        exceptions::CommunicationException,
                        exceptions::AuthorizationException,
                        exceptions::DataValidationException,
                        exceptions::NotAcceptedException,
                        exceptions::TransactionFailedException,
                        exceptions::AlreadyExistsException
                    );

            void acceptRequestForQuote(
                in cmiQuote::RFQEntryStruct rfq )
                    raises(
                        exceptions::SystemException,
                        exceptions::CommunicationException,
                        exceptions::AuthorizationException,
                        exceptions::DataValidationException,
                        exceptions::NotAcceptedException,
                        exceptions::TransactionFailedException
                    );

           cmiOrder::OrderIdStruct acceptStrategyOrder(
               in cmiOrder::OrderEntryStruct anOrder,
```

```
            in cmiOrder::LegOrderEntryStructSequence
legEntryDetails)
                raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::DataValidationException,
                    exceptions::NotAcceptedException,
                    exceptions::TransactionFailedException,
                    exceptions::AlreadyExistsException
                );

        void acceptStrategyOrderUpdateRequest(
            in long currentRemainingQuantity,
            in cmiOrder::OrderEntryStruct updatedOrder,
            in cmiOrder::LegOrderEntryStructSequence
legEntryDetails)
                raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::DataValidationException,
                    exceptions::NotAcceptedException,
                    exceptions::TransactionFailedException
                );

        cmiOrder::OrderIdStruct
acceptStrategyOrderCancelReplaceRequest(
            in cmiOrder::CancelRequestStruct cancelRequest,

            in cmiOrder::OrderIdStruct orderId,
            in long cancelQuantity,

in cmiOrder::OrderEntryStruct newOrder,
            in cmiOrder::LegOrderEntryStructSequence
legEntryDetails)
                raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::DataValidationException,
                    exceptions::NotAcceptedException,
                    exceptions::TransactionFailedException
                );

    };
```

UserPreferenceQuery Interface

No change between Version 2.0 Documentation Only and Version 2.0 Beta 1.

Administrator Interface

No change between Version 2.0 Documentation Only and Version 2.0 Beta 1.

TradingSession Interface

No change between Version 2.0 Documentation Only and Version 2.0 Beta 1.

UserHistory Interface

No change between Version 2.0 Documentation Only and Version 2.0 Beta 1.

UserTradingParameters Interface

No change between Version 2.0 Documentation Only and Version 2.0 Beta 1.

UserSessionManager Interface

No change between Version 2.0 Documentation Only and Version 2.0 Beta 1.

UserAccess Interface

No change between Version 2.0 Documentation Only and Version 2.0 Beta 1.

cmiCallback.idl

No change between Version 2.0 Documentation Only and Version 2.0 Beta 1.

cmiAdmin.idl

No change between Version 2.0 Documentation Only and Version 2.0 Beta 1.

cmiMarketData.idl

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

    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;
    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;

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

```
struct OrderBookSideStruct
    {
        string side;
        cmiMarketData::OrderBookPriceStructSequence
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;
        string exchange;
        cmiMarketData::BookDepthUpdatePriceStructSequence
buySideChanges;
        cmiMarketData::BookDepthUpdatePriceStructSequence
sellSideChanges;
    };

    typedef sequence <BookDepthUpdateStruct>
BookDepthUpdateStructSequence;

}
```

cmiOrder.idl

No change between Version 2.0 Documentation Only and Version 2.0 Beta 1.

cmiProduct.idl

No change between Version 2.0 Documentation Only and Version 2.0 Beta 1.

cmiQuote.idl

No change between Version 2.0 Documentation Only and Version 2.0 Beta 1.

cmiSession.idl

No change between Version 2.0 Documentation Only and Version 2.0 Beta 1.

cmiStrategy.idl

No change between Version 2.0 Documentation Only and Version 2.0 Beta 1.

cmiTraderActivity.idl

No change between Version 2.0 Documentation Only and Version 2.0 Beta 1.

cmiUser.idl

No change between Version 2.0 Documentation Only and Version 2.0 Beta 1.

cmiUtil.idl

No change between Version 2.0 Documentation Only and Version 2.0 Beta 1.

cmiConstants.idl

No change between Version 2.0 Documentation Only and Version 2.0 Beta 1.

cmiErrorCodes.idl

```
interface DataValidationCodes  {
        const exceptions::ErrorCode DUPLICATE_ID = 1000;
        const exceptions::ErrorCode INVALID_VOLUME = 1010;
        const exceptions::ErrorCode INVALID_TIME = 1020;
        const exceptions::ErrorCode INCOMPLETE_QUOTE = 1030;
        const exceptions::ErrorCode INVALID_QUANTITY = 1040;
        const exceptions::ErrorCode INVALID_STRATEGY = 1060;
        const exceptions::ErrorCode INVALID_SPREAD = 1070;
        const exceptions::ErrorCode INVALID_MEMBERKEY = 1080;
        const exceptions::ErrorCode INVALID_PRODUCT = 1090;
```

```
        const exceptions::ErrorCode INVALID_SESSION = 1100;
        const exceptions::ErrorCode INVALID_STATE = 1110;
        const exceptions::ErrorCode PREFERENCE_PATH_MISMATCH =
1120;
        const exceptions::ErrorCode INVALID_ORDER_ID = 1130;
        const exceptions::ErrorCode LISTENER_ALREADY_REGISTERED =
1140;
        const exceptions::ErrorCode INVALID_SIDE = 1150;
        const exceptions::ErrorCode INVALID_PRICE = 1160;
        const exceptions::ErrorCode INVALID_UPDATE_ATTEMPT =
1170;
        const exceptions::ErrorCode INVALID_ORIGINATOR = 1180;
        const exceptions::ErrorCode INVALID_ACCOUNT = 1200;
        const exceptions::ErrorCode INVALID_EXECUTING_GIVEUP_FIRM
= 1210;
        const exceptions::ErrorCode INVALID_CONTINGENCY_TYPE =
1220;
        const exceptions::ErrorCode INVALID_TIME_IN_FORCE = 1230;
        const exceptions::ErrorCode INVALID_POSITION_EFFECT =
1240;
        const exceptions::ErrorCode INVALID_ORIGIN_TYPE = 1250;
        const exceptions::ErrorCode INVALID_COVERAGE = 1260;
        const exceptions::ErrorCode INVALID_PRODUCT_TYPE = 1270;
        const exceptions::ErrorCode INVALID_ORDER_STATE = 1280;
        const exceptions::ErrorCode INVALID_ORDER_SOURCE = 1290;
        const exceptions::ErrorCode
INVALID_BRANCH_SEQUENCE_NUMBER = 1300;
        const exceptions::ErrorCode MISSING_LISTENER = 1310;
        const exceptions::ErrorCode BUSINESS_DAY_NOT_STARTED =
1320;
        const exceptions::ErrorCode INVALID_FIELD_LENGTH = 1330;
        const exceptions::ErrorCode INVALID_STRATEGY_LEG = 1340;
        const exceptions::ErrorCode DUPLICATE_STRATEGY_LEG =
1350;
        const exceptions::ErrorCode INVALID_LEG_CONTINGENCY =
1360;
        const exceptions::ErrorCode INVALID_CANCEL_REQUEST =
1370;
        const exceptions::ErrorCode INVALID_VERSION = 1380;
        const exceptions::ErrorCode INVALID_LOGIN_MODE = 1390;
        const exceptions::ErrorCode
GMD_LISTENER_ALREADY_REGISTERED = 1400;
    };
interface AuthenticationCodes  {
        const exceptions::ErrorCode UNKNOWN_USER = 2000;
        const exceptions::ErrorCode INCORRECT_PASSWORD = 2010;
        const exceptions::ErrorCode FUNCTION_NOT_IMPLEMENTED =
2020;
        const exceptions::ErrorCode INVALID_CLIENT_LOGIN_MODE =
2030;
    };

    interface CommunicationFailureCodes  {
        const exceptions::ErrorCode TRANSPORT_FAILURE = 2500;
```

```
        const exceptions::ErrorCode ROUTING_SESSION_UNAVAILABLE =
2510;
    };

    interface TransactionFailedCodes  {
        const exceptions::ErrorCode CREATE_FAILED = 3000;
        const exceptions::ErrorCode UPDATE_FAILED = 3010;
        const exceptions::ErrorCode ACTION_VETOED = 3020;
        const exceptions::ErrorCode INSUFFICIENT_QUANTITY = 3030;
        const exceptions::ErrorCode REMAINING_QUANTITY_MISMATCH =
3040;
        const exceptions::ErrorCode INVALID_STATE_CHANGE = 3050;
        const exceptions::ErrorCode INCOMPLETE_STATE_CHANGE =
3060;
        const exceptions::ErrorCode NO_REMAINING_QUANTITY = 3070;
    };

    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;
    };

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

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

Summary of Changes to CMi Documentation Only

This section summarizes the changes in the IDL between CBOEdirect Version 1.0 and the CMi Documentation Only release.

cmi.idl

MarketQuery

Added operations to subscribe for book depth:

subscribBookDepth(), unsubscribeBookDepth()

Used to subscribe for dynamic updates of the entire book.

subscribeBookDepthUpdate(), unsubscribeBookDepthUpdated()

Used to subscribe to only the changes that occur in a book. THIS IS AVAILABLE IN THE CMI VERSION 2.0 IDL BUT WILL NOT BE IMPLEMENTED IN CBOEDIRECT 2.0.

Quote

Added operation to subscribe for quote status without publication of current quote status:

subscribeQuoteStatusWithoutPublish()

Deleted operation to subscribe for quote status by class:

subscribeQuoteStatusByClass(), unsubscribeQuoteStatusByClass()

Added parameter to subscription operations that indicates if the callback is to use guaranteed message delivery:

CMi Version 2.0 now supports guaranteed message delivery to your application over the standard CORBA callback object. One consumer callback object for each userid (across all applications currently logged in) may register using guaranteed message delivery.

OrderQuery

Removed requirement to subscribe for order status when querying for orders using the following operations:

getOrdersForProduct(), getOrdersForSession(), getOrdersForType(), getOrdersForClass()

Added ability to for callback objects to use guaranteed message delivery (GMD) using the following subscription operations:

subscribeOrdersByFirm(), subscribeOrdersByFirmWithoutPublish(), subscribeOrders(), subscribeOrdersWithoutPublish()

Added operation to subscribe for order status without publication of status for current orders:

subscribeOrdersWithoutPublish(),
subscribeOrdersByFirmWithoutPublish()

ProductQuery

No change.

ProductDefinition

Added operations to build request for strategy products that can then be used to define strategies using the acceptStrategy() operation:

buildStrategyRequestByName() – permits you to specify a strategy type (such as vertical or time), specify an anchor product by name and a price and month increment for the subsequent legs of the strategy.

buildStrategyRequestByProductKey() – same as buildStrategyRequestByName() except uses a CBOEdirect ProductKey instead of a product name.

OrderEntry

Added operations to enter and modify strategy orders:

acceptStrategyOrder(), acceptStrategyOrderUpdateRequest(), acceptStrategyOrderCancelReplaceRequest()

UserPreferenceQuery

No change

Administrator

No change

TradingSession

Removed ability to retrieve all class and product information for a trading session:

getClassDetailForTradingSession() is no longer supported. This operation was removed to force users to retrieve product information in a more granular manner in anticipation of support for product information for floor based products.

Strategy retrieval operations now supported:

getStrategiesByClassForSession() and getStrategyBySessionForKey() are now implemented to support trading strategies using CBOEdirect.

Added operation to retrieve all strategies in a trading session that contain a particular product as a leg:

getStrategiesByComponent()

UserHistory

No change

UserTradingParameters

No change

UserSessionManager

No change

UserAccess

Added capability to subscribe for text messages using guaranteed message delivery (GMD).

logon()

cmiCallBack.idl

Added callback object to receive a new image of the order book dynamically:

CMIOrderBookConsumer

Added callback object to receive order book updates dynamically:

CMIOrderBookUpdateConsumer

NOTE: This Callback is provided, but not supported by CBOEdirect 2.0.

cmiAdmin.idl

No change.

cmiMarketData.idl

No change.

cmiOrder.idl

Order Status Reports have been modified to support reporting of strategy legs:

A new LegOrderEntryStruct has been added to specify information, such as open/close and covered/uncovered per leg for a strategy order.

The order messages now support the notion of multiple exchange identifiers in anticipation of single stock futures.

Each report message can now contain multiple reports to support reporting for legs of strategies.

Each report (fill, cancel, bust) has an indicator for the type of report (single leg product, overall strategy, leg of a strategy).

cmiProduct.idl

The ClassStruct now includes an attribute that indicates if the class is a test class – to support test classes in various environments (such as production).

cmiQuote.idl

The quote fill and bust reports now contain multiple reports to support reporting of legs of strategies.

cmiSession.idl

Added SessionStrategyLegStruct and SessionStrategyStruct to support trading strategies within a trading session.

cmiStrategy.idl

Introduced StrategyTypes to ease definition of strategies.

cmiTraderActivity.idl

No change.

cmiUser.idl

Added ExchangeFirmStruct that contains an exchange and firm number at that exchange in anticipation of the single stock futures joint venture.

cmiUtil.idl

Added a report type that is used to indicate the type of report – single product, overal strategy, or leg of a strategy.

cmiConstants.idl

Added constants to the cmiConstants::Sides to support strategy trading.

Added StrategyTypes interface that contains enumerations for strategy types that can be used to define strategies within CBOEdirect, such as Straddle, vertical, ratio, etc.

cmiErrorCodes.idl

No change.

IDL Changes for Documentation Only

cmi.idl

IDL Changes are shown as follows:

Code that is unchanged from the previous version is shown in regular text:

```
interface MarketQuery
    {
        void subscribeRecapForClass(
            in cmiSession::TradingSessionName sessionName,
            in cmiProduct::ClassKey classKey,
            in cmiCallback::CMIRecapConsumer clientListener)
                raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::DataValidationException
                );
```

IDL that has been added is shown in red inside a text box:

```
void aNewMethodName(
            in cmiSession::TradingSessionName sessionName,
            in cmiProduct::ProductKey productKey,
            in cmiCallback::CMIOrderBookConsumer
orderBookConsumer)
                raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::DataValidationException,
                    exceptions::NotFoundException
            );
```

IDL that has been deleted is shown in grey in strike through inside a text box:

```
void someDeletedOperation(
            in cmiSession::TradingSessionName sessionName,
            in cmiProduct::ProductKey productKey,
            in cmiCallback::CMIOrderBookConsumer
orderBookConsumer)
                raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::DataValidationException,
                    exceptions::NotFoundException
            );
```

MarketQuery Interface

```
interface MarketQuery
      {
```

```
void subscribeRecapForClass(
      in cmiSession::TradingSessionName sessionName,
      in cmiProduct::ClassKey classKey,
      in cmiCallback::CMIRecapConsumer clientListener)
      raises(
            exceptions::SystemException,
            exceptions::CommunicationException,
            exceptions::AuthorizationException,
            exceptions::DataValidationException
            );

void subscribeRecapForProduct(
      in cmiSession::TradingSessionName sessionName,
      in cmiProduct::ProductKey productKey,
      in cmiCallback::CMIRecapConsumer clientListener)
      raises(
            exceptions::SystemException,
            exceptions::CommunicationException,
            exceptions::AuthorizationException,
            exceptions::DataValidationException
            );
void subscribeCurrentMarketForClass(
      in cmiSession::TradingSessionName sessionName,
      in cmiProduct::ClassKey classKey,
      in cmiCallback::CMICurrentMarketConsumer
      clientListener)
      raises(
            exceptions::SystemException,
            exceptions::CommunicationException,
            exceptions::AuthorizationException,
            exceptions::DataValidationException
                  );
void subscribeCurrentMarketForProduct(
      in cmiSession::TradingSessionName sessionName,
      in cmiProduct::ProductKey productKey,
      in cmiCallback::CMICurrentMarketConsumer
      clientListener)
      raises(
            exceptions::SystemException,
            exceptions::CommunicationException,
            exceptions::AuthorizationException,
            exceptions::DataValidationException
            );
void subscribeNBBOForClass(
      in cmiSession::TradingSessionName sessionName,
      in cmiProduct::ClassKey classKey,
      in cmiCallback::CMINBBOConsumer clientListener)
      raises(
            exceptions::SystemException,
            exceptions::CommunicationException,
            exceptions::AuthorizationException,
            exceptions::DataValidationException
            );
```

```
void subscribeNBBOForProduct(
      in cmiSession::TradingSessionName sessionName,
      in cmiProduct::ProductKey productKey,
      in cmiCallback::CMINBBOConsumer clientListener)
      raises(
            exceptions::SystemException,
            exceptions::CommunicationException,
            exceptions::AuthorizationException,
            exceptions::DataValidationException
            );

void unsubscribeCurrentMarketForProduct(
      in cmiSession::TradingSessionName sessionName,
      in cmiProduct::ProductKey productKey,
      in cmiCallback::CMICurrentMarketConsumer
      clientListener)
      raises(
            exceptions::SystemException,
            exceptions::CommunicationException,
            exceptions::AuthorizationException,
            exceptions::DataValidationException
            );

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

void unsubscribeNBBOForProduct(
      in cmiSession::TradingSessionName sessionName,
      in cmiProduct::ProductKey productKey,
      in cmiCallback::CMINBBOConsumer clientListener)
      raises(
            exceptions::SystemException,
            exceptions::CommunicationException,
            exceptions::AuthorizationException,
            exceptions::DataValidationException
            );
void unsubscribeNBBOForClass(
      in cmiSession::TradingSessionName sessionName,
      in cmiProduct::ClassKey classKey,
      in cmiCallback::CMINBBOConsumer clientListener)
      raises(
            exceptions::SystemException,
            exceptions::CommunicationException,
            exceptions::AuthorizationException,
            exceptions::DataValidationException
            );
```

```
void subscribeTicker(
      in cmiSession::TradingSessionName sessionName,
      in cmiProduct::ProductKey productKey,
      in cmiCallback::CMITickerConsumer clientListener)
      raises(
            exceptions::SystemException,
            exceptions::CommunicationException,
            exceptions::AuthorizationException,
            exceptions::DataValidationException
                  );
void unsubscribeRecapForProduct(
      in cmiSession::TradingSessionName sessionName,
      in cmiProduct::ProductKey productKey,
      in cmiCallback::CMIRecapConsumer clientListener)
      raises(
            exceptions::SystemException,
            exceptions::CommunicationException,
            exceptions::AuthorizationException,
            exceptions::DataValidationException
            );

void unsubscribeRecapForClass(
      in cmiSession::TradingSessionName sessionName,
      in cmiProduct::ClassKey classKey,
      in cmiCallback::CMIRecapConsumer clientListener)
      raises(
            exceptions::SystemException,
            exceptions::CommunicationException,
            exceptions::AuthorizationException,
            exceptions::DataValidationException
            );

void unsubscribeTicker(
      in cmiSession::TradingSessionName sessionName,
      in cmiProduct::ProductKey productKey,
      in cmiCallback::CMITickerConsumer clientListener)
      raises(
            exceptions::SystemException,
            exceptions::CommunicationException,
            exceptions::AuthorizationException,
            exceptions::DataValidationException
            );
            exceptions::SystemException,
            exceptions::CommunicationException,
            exceptions::AuthorizationException,
            exceptions::DataValidationException,
            exceptions::NotFoundException
            );

      void unsubscribeBookDepth(
            in cmiSession::TradingSessionName sessionName,
            in cmiProduct::ProductKey productKey,
```

```
in cmiCallback::CMIOrderBookConsumer
orderBookConsumer)
raises(
      exceptions::SystemException,
      exceptions::CommunicationException,
      exceptions::AuthorizationException,
      exceptions::DataValidationException,
      exceptions::NotFoundException
      );

void subscribeBookDepthUpdate(
      in cmiSession::TradingSessionName sessionName,
      in cmiProduct::ProductKey productKey,
      in cmiCallback::CMIOrderBookUpdateConsumer
      orderBookConsumer)
      raises(
      exceptions::SystemException,
      exceptions::CommunicationException,
      exceptions::AuthorizationException,
      exceptions::DataValidationException,
      exceptions::NotFoundException
      );

void unsubscribeBookDepthUpdate(
      in cmiSession::TradingSessionName sessionName,
      in cmiProduct::ProductKey productKey,
      in cmiCallback::CMIOrderBookUpdateConsumer
      orderBookConsumer)
      raises(
      exceptions::SystemException,
      exceptions::CommunicationException,
      exceptions::AuthorizationException,
      exceptions::DataValidationException,
      exceptions::NotFoundException
      );
                              };
```

Quote Interface

```
interface Quote {
      void acceptQuote(
            in cmiQuote::QuoteEntryStruct quote )
            raises(
                  exceptions::SystemException,
                  exceptions::CommunicationException,
                  exceptions::AuthorizationException,
                  exceptions::DataValidationException,
                  exceptions::NotAcceptedException,
                  exceptions::TransactionFailedException
```

```
                        );

                cmiQuote::ClassQuoteResultStructSequence

        acceptQuotesForClass(
                in cmiProduct::ClassKey classKey,
                in cmiQuote::QuoteEntryStructSequence quotes )
                raises(
                        exceptions::SystemException,
                        exceptions::CommunicationException,
                        exceptions::AuthorizationException,
                        exceptions::DataValidationException,
                        exceptions::NotAcceptedException,
                        exceptions::TransactionFailedException
                        );

        cmiQuote::QuoteDetailStruct getQuote(
                in cmiSession::TradingSessionName sessionName,
                in cmiProduct::ProductKey productKey)
                raises(
                        exceptions::SystemException,
                        exceptions::CommunicationException,
                        exceptions::AuthorizationException,
                        exceptions::DataValidationException,
                        exceptions::NotFoundException
                        );

        void subscribeQuoteStatus(
                in cmiCallback::CMIQuoteStatusConsumer
                clientListener,
                in boolean gmdCallback)
                raises(
                        exceptions::SystemException,
                        exceptions::CommunicationException,
                        exceptions::DataValidationException,
                        exceptions::AuthorizationException
                        );

        void subscribeQuoteStatusWithoutPublish(
                in cmiCallback::CMIQuoteStatusConsumer
                clientListener,
                in boolean gmdCallback)
                raises(
                        exceptions::SystemException,
                        exceptions::CommunicationException,
                        exceptions::DataValidationException,
                        exceptions::AuthorizationException
                        );

        void unsubscribeQuoteStatus(
                in cmiCallback::CMIQuoteStatusConsumer
                clientListener)
                raises(
                        exceptions::SystemException,
```

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

      void subscribeQuoteStatusForFirm(
            in cmiCallback::CMIQuoteStatusConsumer
            clientListener,
            in boolean gmdCallback)
            raises(
                  exceptions::SystemException,
                  exceptions::CommunicationException,
                  exceptions::DataValidationException,
                  exceptions::AuthorizationException
                  );

      void subscribeQuoteStatusForFirmWithoutPublish(
            in cmiCallback::CMIQuoteStatusConsumer
            clientListener,
            in boolean gmdCallback)
            raises(
                  exceptions::SystemException,
                  exceptions::CommunicationException,
                  exceptions::DataValidationException,
                  exceptions::AuthorizationException
                  );

      void unsubscribeQuoteStatusForFirm(
            in cmiCallback::CMIQuoteStatusConsumer
            clientListener)
            raises(
                  exceptions::SystemException,
                  exceptions::CommunicationException,
                  exceptions::AuthorizationException,
                  exceptions::DataValidationException
                  );

      void cancelQuote(
            in cmiSession::TradingSessionName sessionName,
            in cmiProduct::ProductKey productKey)
            raises(
                  exceptions::SystemException,
                  exceptions::CommunicationException,

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

      void cancelAllQuotes(
            in cmiSession::TradingSessionName sessionName)
            raises(
                  exceptions::SystemException,
```

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

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

        void subscribeQuoteStatusByClass(
              in cmiProduct::ClassKey classKey,
              in cmiSession::TradingSessionName sessionName,
              in cmiCallback::CMIQuoteStatusConsumer
              clientListener)
              raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::DataValidationException
                    );
        void unsubscribeQuoteStatusByClass(
              in cmiProduct::ClassKey classKey,
              in cmiSession::TradingSessionName sessionName,
              in cmiCallback::CMIQuoteStatusConsumer clientListener
              )
              raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::DataValidationException
                    );

        void subscribeRFQ(
              in cmiSession::TradingSessionName sessionName,
              in cmiProduct::ClassKey classKey,
              in cmiCallback::CMIRFQConsumer clientListener )
              raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::DataValidationException
                    );

        void unsubscribeRFQ(
```

```
                in cmiSession::TradingSessionName sessionName,
                in cmiProduct::ClassKey classKey,
                in cmiCallback::CMIRFQConsumer clientListener)
                raises(
                      exceptions::SystemException,
                      exceptions::CommunicationException,
                      exceptions::AuthorizationException,
                      exceptions::DataValidationException
                      );
```

OrderQuery Interface

```
                cmiOrder::OrderDetailStructSequence getOrdersForProduct(
                in cmiProduct::ProductKey productKey)
                in cmiCallback::CMIOrderStatusConsumer
                clientListener)
                raises(
                      exceptions::SystemException,
                      exceptions::CommunicationException,
                      exceptions::AuthorizationException,
                      exceptions::DataValidationException
                      );

                cmiOrder::OrderDetailStructSequence getOrdersForSession (
                in cmiSession::TradingSessionName sessionName)
                in cmiCallback::CMIOrderStatusConsumer
                clientListener)
                raises(
                      exceptions::SystemException,
                      exceptions::CommunicationException,
                      exceptions::AuthorizationException,
                      exceptions::DataValidationException
                      );

                cmiOrder::OrderDetailStruct getOrderById(
                in cmiOrder::OrderIdStruct orderId)
                raises(
                      exceptions::SystemException,
                      exceptions::CommunicationException,
                      exceptions::AuthorizationException,
                      exceptions::DataValidationException,
                      exceptions::NotFoundException
                );

          void subscribeOrdersByFirm(
                in cmiCallback::CMIOrderStatusConsumer
                clientListener,
                in boolean gmdCallback)
                raises(
                      exceptions::SystemException,
                      exceptions::CommunicationException,
                      exceptions::AuthorizationException,
                      exceptions::DataValidationException
                      );
```

```
void subscribeOrdersByFirmWithoutPublish(
      in cmiCallback::CMIOrderStatusConsumer
      clientListener,
      in boolean gmdCallback)
      raises(
            exceptions::SystemException,
            exceptions::CommunicationException,
            exceptions::AuthorizationException,
            exceptions::DataValidationException
            );

void subscribeOrders(
      in cmiCallback::CMIOrderStatusConsumer
      clientListener,
      in boolean gmdCallback)
      raises(
            exceptions::SystemException,
            exceptions::CommunicationException,
            exceptions::AuthorizationException,
            exceptions::DataValidationException
            );

void subscribeOrdersWithoutPublish(
      in cmiCallback::CMIOrderStatusConsumer
      clientListener,
      in boolean gmdCallback)
      raises(
            exceptions::SystemException,
            exceptions::CommunicationException,
            exceptions::AuthorizationException,
            exceptions::DataValidationException
            );

cmiOrder::OrderDetailStructSequence getOrdersForType(
      in cmiProduct::ProductType productType)
      in cmiCallback::CMIOrderStatusConsumer
      clientListener)
      raises(
            exceptions::SystemException,
            exceptions::CommunicationException,
            exceptions::AuthorizationException,
            exceptions::DataValidationException
            );

cmiOrder::OrderDetailStructSequence getOrdersForClass(
      in cmiProduct::ClassKey classKey)
      in cmiCallback::CMIOrderStatusConsumer
      clientListener)
      raises(
            exceptions::SystemException,
            exceptions::CommunicationException,
            exceptions::AuthorizationException,
```

```
                  exceptions::DataValidationException
                  );

      cmiTraderActivity::ActivityHistoryStruct
            queryOrderHistory(
            in cmiOrder::OrderIdStruct orderId )
            raises(
                  exceptions::SystemException,
                  exceptions::CommunicationException,
                  exceptions::AuthorizationException,
                  exceptions::NotFoundException,
                  exceptions::DataValidationException
                  );

      cmiOrder::PendingOrderStructSequence
            getPendingAdjustmentOrdersByProduct(
            in cmiSession::TradingSessionName sessionName,
            in cmiProduct::ProductKey productKey)
            raises(
                  exceptions::SystemException,
                  exceptions::CommunicationException,
                  exceptions::AuthorizationException,
                  exceptions::DataValidationException
                  );

      cmiOrder::PendingOrderStructSequence
            getPendingAdjustmentOrdersByClass(
            in cmiSession::TradingSessionName sessionName,
            in cmiProduct::ClassKey classKey)
            raises(
                  exceptions::SystemException,
                  exceptions::CommunicationException,
                  exceptions::AuthorizationException,
                  exceptions::DataValidationException
                  );

      void unsubscribeOrderStatusForProduct(
            in cmiProduct::ProductKey productKey,
            in cmiCallback::CMIOrderStatusConsumer
            clientListener)
            raises(
                  exceptions::SystemException,
                  exceptions::CommunicationException,
                  exceptions::AuthorizationException,
                  exceptions::DataValidationException
                  );

      void unsubscribeOrderStatusForSession(
            in cmiSession::TradingSessionName sessionName,
            in cmiCallback::CMIOrderStatusConsumer
            clientListener)
            raises(
                  exceptions::SystemException,
                  exceptions::CommunicationException,
                  exceptions::AuthorizationException,
```

```
                exceptions::DataValidationException
                );

        void unsubscribeOrderStatusForFirm(
              in cmiCallback::CMIOrderStatusConsumer
              clientListener)
              raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::DataValidationException
                    );

        void unsubscribeAllOrderStatusForType(
              in cmiProduct::ProductType productType,
              in cmiCallback::CMIOrderStatusConsumer
              clientListener)
              raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::DataValidationException
                    );

        void unsubscribeOrderStatusByClass(
              in cmiProduct::ClassKey classKey,
              in cmiCallback::CMIOrderStatusConsumer
              clientListener)
              raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::DataValidationException
                    );
                          };
```

ProductQuery Interface

No changes between versions 1.0b and 2.0.

ProductDefinition Interface

```
        interface ProductDefinition {
              cmiSession::SessionStrategyStruct acceptStrategy(
              in cmiSession::TradingSessionName sessionName,
              in cmiStrategy::StrategyRequestStruct strategyRequest
              )
              raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
```

```
                exceptions::DataValidationException
                );

        cmiStrategy::StrategyRequestStruct
                buildStrategyRequestByName(
                in cmiStrategy::StrategyType strategyType,
                in cmiProduct::ProductNameStruct anchorProduct,
                in cmiUtil::PriceStruct priceIncrement,
                in cmiStrategy::MonthIncrement monthIncrement)
                raises(
                        exceptions::SystemException,
                        exceptions::DataValidationException,
                        exceptions::AuthorizationException,
                        exceptions::CommunicationException
                        );

        cmiStrategy::StrategyRequestStruct
                buildStrategyRequestByProductKey(
                in cmiStrategy::StrategyType strategyType,
                in cmiProduct::ProductKey anchorProductKey,
                in cmiUtil::PriceStruct priceIncrement,
                in cmiStrategy::MonthIncrement monthIncrement)
                raises(
                        exceptions::SystemException,
                        exceptions::DataValidationException,
                        exceptions::AuthorizationException,
                        exceptions::CommunicationException
                        );
};
```

OrderEntry Interface

```
interface OrderEntry {
      cmiOrder::OrderIdStruct acceptOrder(
            in cmiOrder::OrderEntryStruct anOrder)
            raises(
                  exceptions::SystemException,
                  exceptions::CommunicationException,
                  exceptions::AuthorizationException,
                  exceptions::DataValidationException,
                  exceptions::NotAcceptedException,
                  exceptions::TransactionFailedException,
                  exceptions::AlreadyExistsException
                  );

      cmiOrder::OrderIdStruct acceptOrderByProductName(
            in cmiProduct::ProductNameStruct product,
            in cmiOrder::OrderEntryStruct anOrder)
            raises(
                  exceptions::SystemException,
                  exceptions::CommunicationException,
                  exceptions::AuthorizationException,
```

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

      void acceptOrderCancelRequest(
            in cmiOrder::CancelRequestStruct cancelRequest)
            raises(
                  exceptions::SystemException,
                  exceptions::CommunicationException,
                  exceptions::AuthorizationException,
                  exceptions::DataValidationException,
                  exceptions::NotAcceptedException,
                  exceptions::TransactionFailedException
                  );

      void acceptOrderUpdateRequest(
            in long currentRemainingQuantity,
            in cmiOrder::OrderEntryStruct updatedOrder)
            raises(
                  exceptions::SystemException,
                  exceptions::CommunicationException,
                  exceptions::AuthorizationException,
                  exceptions::DataValidationException,
                  exceptions::NotAcceptedException,
                  exceptions::TransactionFailedException
                  );

      cmiOrder::OrderIdStruct acceptOrderCancelReplaceRequest(
            in cmiOrder::OrderIdStruct orderId,
            in long cancelQuantity,
            raises(
                  exceptions::SystemException,
                  exceptions::CommunicationException,
                  exceptions::AuthorizationException,
                  exceptions::DataValidationException,
                  exceptions::NotAcceptedException,
                  exceptions::TransactionFailedException
                  );


      void acceptCrossingOrder(
            in cmiOrder::OrderEntryStruct buyCrossingOrder,
            in cmiOrder::OrderEntryStruct sellCrossingOrder)
            raises(
                  exceptions::SystemException,
                  exceptions::CommunicationException,
                  exceptions::AuthorizationException,
                  exceptions::DataValidationException,
                  exceptions::NotAcceptedException,
                  exceptions::TransactionFailedException,
                  exceptions::AlreadyExistsException
                  );
```

```
void acceptRequestForQuote(
      in cmiQuote::RFQEntryStruct rfq )
      raises(
            exceptions::SystemException,
            exceptions::CommunicationException,
            exceptions::AuthorizationException,
            exceptions::DataValidationException,
            exceptions::NotAcceptedException,
            exceptions::TransactionFailedException
            );
```

```
cmiOrder::OrderIdStruct acceptStrategyOrder(
      in cmiOrder::OrderEntryStruct anOrder,
      in cmiOrder::LegOrderEntryStructSequence
      legEntryDetails)
      raises(
            exceptions::SystemException,
            exceptions::CommunicationException,
            exceptions::AuthorizationException,
            exceptions::DataValidationException,
            exceptions::NotAcceptedException,
            exceptions::TransactionFailedException,
            exceptions::AlreadyExistsException
            );
```

```
void acceptStrategyOrderUpdateRequest(
      in long currentRemainingQuantity,
      in cmiOrder::OrderEntryStruct updatedOrder,
      in cmiOrder::LegOrderEntryStructSequence
      legEntryDetails)
      raises(
            exceptions::SystemException,
            exceptions::CommunicationException,
            exceptions::AuthorizationException,
            exceptions::DataValidationException,
            exceptions::NotAcceptedException,
            exceptions::TransactionFailedException
            );
cmiOrder::OrderIdStruct
      acceptStrategyOrderCancelReplaceRequest(
      in cmiOrder::OrderIdStruct orderId,
      in long cancelQuantity,
      in cmiOrder::OrderEntryStruct newOrder,
      in cmiOrder::LegOrderEntryStructSequence
      legEntryDetails)
      raises(
            exceptions::SystemException,
            exceptions::CommunicationException,
            exceptions::AuthorizationException,
            exceptions::DataValidationException,
            exceptions::NotAcceptedException,
            exceptions::TransactionFailedException
```

```
            );
};
```

UserPreferenceQuery Interface

No changes between version 1.0b and 2.0.

Administrator Interface

No changes between version 1.0b and 2.0.

TradingSession Interface

```
interface TradingSession
            {
      cmiSession::TradingSessionStructSequence
            getCurrentTradingSessions(
            in cmiCallback::CMITradingSessionStatusConsumer
            clientListener)
            raises(
                  exceptions::SystemException,
                  exceptions::CommunicationException,
                  exceptions::AuthorizationException,
                  exceptions::DataValidationException
                  );

      void unsubscribeTradingSessionStatus(
            in cmiCallback::CMITradingSessionStatusConsumer
            clientListener)
            raises(
                  exceptions::SystemException,
                  exceptions::CommunicationException,
                  exceptions::AuthorizationException,
                  exceptions::DataValidationException
                  );

      cmiSession::SessionClassDetailStructSequence
            getClassDetailForTradingSession(
            in cmiSession::TradingSessionName sessionName)
            raises(
                  exceptions::SystemException,
                  exceptions::CommunicationException,
                  exceptions::AuthorizationException,
                  exceptions::DataValidationException
                  );

      cmiProduct::ProductTypeStructSequence
            getProductTypesForSession(
            in cmiSession::TradingSessionName sessionName)
            raises(
                  exceptions::SystemException,
                  exceptions::CommunicationException,
                  exceptions::AuthorizationException,
```

```
                  exceptions::DataValidationException
                  );

      cmiSession::SessionClassStructSequence
            getClassesForSession(
            in cmiSession::TradingSessionName sessionName,
            in cmiProduct::ProductType productType,
            in cmiCallback::CMIClassStatusConsumer
            clientListener)
            raises(
                  exceptions::SystemException,
                  exceptions::CommunicationException,
                  exceptions::AuthorizationException,
                  exceptions::DataValidationException
                  );

      cmiSession::SessionProductStructSequence
            getProductsForSession(
            in cmiSession::TradingSessionName sessionName,
            in cmiProduct::ClassKey classKey,
            in cmiCallback::CMIProductStatusConsumer
            clientListener)
            raises(
                  exceptions::SystemException,
                  exceptions::CommunicationException,
                  exceptions::AuthorizationException,
                  exceptions::DataValidationException
                  );

      cmiSession::SessionStrategyStructSequence
            getStrategiesByClassForSession(
            in cmiSession::TradingSessionName sessionName,
            in cmiProduct::ClassKey classKey,
            in cmiCallback::CMIStrategyStatusConsumer
            clientListener)
            raises(
                  exceptions::SystemException,
                  exceptions::CommunicationException,
                  exceptions::AuthorizationException,
                  exceptions::DataValidationException
                  );

      cmiSession::SessionProductStruct
            getProductBySessionForKey(
            in cmiSession::TradingSessionName sessionName,
            in cmiProduct::ProductKey productKey)
            raises(
                  exceptions::SystemException,
                  exceptions::CommunicationException,
                  exceptions::AuthorizationException,
                  exceptions::DataValidationException,
                  exceptions::NotFoundException
                  );

      cmiSession::SessionStrategyStruct
```

```
            getStrategyBySessionForKey(
            in cmiSession::TradingSessionName sessionName,
            in cmiProduct::ProductKey productKey)
            raises(
                  exceptions::SystemException,
                  exceptions::CommunicationException,
                  exceptions::AuthorizationException,
                  exceptions::DataValidationException,
                  exceptions::NotFoundException
                  );

      cmiSession::SessionClassStruct getClassBySessionForKey(
            in cmiSession::TradingSessionName sessionName,
            in cmiProduct::ClassKey classKey)
            raises(
                  exceptions::SystemException,
                  exceptions::CommunicationException,
                  exceptions::AuthorizationException,
                  exceptions::DataValidationException,
                  exceptions::NotFoundException
                  );

      cmiSession::SessionProductStruct
            getProductBySessionForName(
            in cmiSession::TradingSessionName sessionName,
            in cmiProduct::ProductNameStruct productName )
            raises(
                  exceptions::SystemException,
                  exceptions::CommunicationException,
                  exceptions::AuthorizationException,
                  exceptions::DataValidationException,
                  exceptions::NotFoundException
                  );

      cmiSession::SessionClassStruct
            getClassBySessionForSymbol(
            in cmiSession::TradingSessionName sessionName,
            in cmiProduct::ProductType productType,
            in string className)
            raises(
                  exceptions::SystemException,
                  exceptions::CommunicationException,
                  exceptions::AuthorizationException,
                  exceptions::DataValidationException,
                  exceptions::NotFoundException
                  );

      void unsubscribeClassesByTypeForSession(
            in cmiSession::TradingSessionName sessionName,
            in cmiProduct::ProductType productType,
            in cmiCallback::CMIClassStatusConsumer
            clientListener)
            raises(
                  exceptions::SystemException,
                  exceptions::CommunicationException,
```

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

        void unsubscribeProductsByClassForSession(
              in cmiSession::TradingSessionName sessionName,
              in cmiProduct::ClassKey classKey,
              in cmiCallback::CMIProductStatusConsumer
              clientListener)
              raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::DataValidationException
                    );

        void unsubscribeStrategiesByClassForSession(
              in cmiSession::TradingSessionName sessionName,
              in cmiProduct::ClassKey classKey,
              in cmiCallback::CMIStrategyStatusConsumer
              clientListener)
              raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::DataValidationException
                    );

  cmiSession::SessionStrategyStructSequence
        getStrategiesByComponent(
        in cmiProduct::ProductKey componentKey,
        in cmiSession::TradingSessionName sessionName)
        raises(
              exceptions::SystemException,
              exceptions::DataValidationException,
              exceptions::AuthorizationException,
              exceptions::CommunicationException
              );
              };
```

UserHistory Interface

No changes between versions 1.0b and 2.0.

UserTradingParameters Interface

No changes between versions 1.0b and 2.0.

UserSessionManager Interface

No changes between versions 1.0b and 2.0.

UserAccess Interface

```
interface UserAccess
            {
      UserSessionManager 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
                  );
            };
```

cmiCallback.idl

```
module cmiCallback
            {
      interface CMICurrentMarketConsumer {
            void acceptCurrentMarket(
            in cmiMarketData::CurrentMarketStructSequence
            currentMarket);
            };

      interface CMINBBOConsumer {
            void acceptNBBO( in cmiMarketData::NBBOStructSequence
            nbbo);
            };

      interface CMIExpectedOpeningPriceConsumer {
            void acceptExpectedOpeningPrice(
            in cmiMarketData::ExpectedOpeningPriceStruct
            expectedOpeningPrice );
            };
      interface CMIOrderStatusConsumer {
            void acceptOrderStatus( in
            cmiOrder::OrderDetailStructSequence orders);

            void acceptOrderCanceledReport(
            in cmiOrder::OrderCancelReportStruct canceledReport
            );

            void acceptOrderFilledReport(
            in cmiOrder::OrderFilledReportStruct filledReport );

            void acceptOrderBustReport(
            in cmiOrder::OrderBustReportStruct bustReport);
```

```
            void acceptOrderBustReinstateReport(
            in cmiOrder::OrderBustReinstateReportStruct
            bustReinstatedReport);

            void acceptNewOrder(in cmiOrder::OrderDetailStruct
            order);
            };

      interface CMIQuoteStatusConsumer {

            void acceptQuoteStatus( in
            cmiQuote::QuoteDetailStructSequence quotes);

            void acceptQuoteFilledReport(
            in cmiQuote::QuoteFilledReportStruct filledReport );

            void acceptQuoteBustReport(
            in cmiQuote::QuoteBustReportStruct bustReport );
      };
      interface CMIRFQConsumer {
            void acceptRFQ(
            in cmiQuote::RFQStruct rfq);
            };

      interface CMIClassStatusConsumer {
            void updateProductClass(in cmiSession::SessionClassStruct
            updatedClass);

            void acceptClassState(in
            cmiSession::ClassStateStructSequence classState);
            };

      interface CMIStrategyStatusConsumer {
            void updateProductStrategy(in
            cmiSession::SessionStrategyStructSequence updatedStrategies);
            };

      interface CMIProductStatusConsumer {
            void updateProduct(in cmiSession::SessionProductStruct
            updatedProduct);

            void acceptProductState(in
            cmiSession::ProductStateStructSequence productState);
            };


      interface CMITradingSessionStatusConsumer {
            void acceptTradingSessionState( in
            cmiSession::TradingSessionStateStruct sessionState);
            };

      interface CMIRecapConsumer {
            void acceptRecap(
            in cmiMarketData::RecapStructSequence recap);
            };
```

```
interface CMITickerConsumer {
      void acceptTicker(
      in cmiMarketData::TickerStructSequence ticker);
      };

interface CMIUserSessionAdmin {
      cmiAdmin::HeartBeatStruct acceptHeartBeat (in
      cmiAdmin::HeartBeatStruct heartbeat);

      void acceptLogout( in string reason );
      void acceptTextMessage( in cmiAdmin::MessageStruct
      message );
      void acceptAuthenticationNotice();
      void acceptCallbackRemoval( in
      cmiUtil::CallbackInformationStruct callbackInformation,
      in string reason,
            in exceptions::ErrorCode
            errorCode);
      };
```

```
interface CMIOrderBookConsumer {
      void acceptTopOfBook(in cmiMarketData::BookDepthStruct
      productBook);
      };
```

```
interface CMIOrderBookUpdateConsumer {
      void acceptBookDepthUpdate(in
      cmiMarketData::BookDepthUpdateStruct productBook );
      };
      };
```

cmiAdmin.idl

No changes between versions 1.0b and 2.0.

CmiMarketData.idl

```
module cmiMarketData
            {
            typedef short ExpectedOpeningPriceType;
            typedef short MarketDataHistoryEntryType;
            typedef short MarketChangeReason;
            typedef short VolumeType;

      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 OrderBookPriceStruct
{
cmiUtil::PriceStruct price;
long marketMakerVolume;
long customerVolume;
long brokerDealerVolume;
long contingencyVolume;
};
typedef sequence <OrderBookPriceStruct>
OrderBookPriceStructSequence;
```

```
struct OrderBookSideStruct
{
string side;
cmiMarketData::OrderBookPriceStructSequence
orderBookPriceStructSequence;
};
struct BookDepthStruct
{
cmiProduct::ProductKey productKey;
cmiSession::TradingSessionName sessionName;
cmiMarketData::OrderBookSideStruct buySideStruct;
cmiMarketData::OrderBookSideStruct sellSideStruct;
};
```

```
struct RecapStruct
{
cmiProduct::ProductKeysStruct productKeys;
cmiSession::TradingSessionName sessionName;
cmiProduct::ProductNameStruct productInformation;
cmiUtil::PriceStruct lastSalePrice;
cmiUtil::TimeStruct tradeTime;
long lastSaleVolume;
long totalVolume;
char tickDirection;
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;

        struct OrderBookPriceStruct
        {
```

```
cmiUtil::PriceStruct price;
long totalVolume;
long contingencyVolume;
};
typedef sequence <OrderBookPriceStruct>
OrderBookPriceStructSequence;
struct OrderBookSideStruct
{
string side;
cmiMarketData::OrderBookPriceStructSequence
orderBookPriceStructSequence;
};
struct BookDepthStruct
{
cmiProduct::ProductKeysStruct productKeys;
cmiSession::TradingSessionName sessionName;
cmiMarketData::OrderBookPriceStructSequence
buySideSequence;
cmiMarketData::OrderBookPriceStructSequence
sellSideSequence;
boolean allPricesIncluded;
long transactionSequenceNumber;
};
typedef sequence <BookDepthStruct> BookDepthStructSequence;
typedef char BookDepthUpdateType;
struct BookDepthUpdatePriceStruct {
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;
string exchange;
BookDepthUpdatePriceStructSequence buySideChanges;
BookDepthUpdatePriceStructSequence sellSideChanges;
};
typedef sequence <BookDepthUpdateStruct>
BookDepthUpdateStructSequence;
};
```

cmiOrder.idl

```
module cmiOrder
{
```

```
typedef short ContingencyType;
typedef short OrderState;
typedef char TimeInForce;
typedef char PositionEffect;
typedef char OriginType;
typedef char Coverage;
typedef boolean CrossingIndicator;
typedef short CancelType;
struct OrderContingencyStruct
{
cmiOrder::ContingencyType type;
cmiUtil::PriceStruct price;
long volume;
};
struct OrderIdStruct
{
cmiUser::ExchangeFirmStruct executingOrGiveUpFirm;

string branch;
long branchSequenceNumber;
string correspondentFirm;
string orderDate; // YYYYMMDD format
long highCboeId;
long lowCboeId;
};
struct OrderEntryStruct
{
cmiUser::ExchangeFirmStruct executingOrGiveUpFirm;

string branch;
long branchSequenceNumber;
string correspondentFirm;
string orderDate; // YYYYMMDD format

cmiUser::ExchangeAcronymStruct originator;

long originalQuantity;
cmiProduct::ProductKey productKey;
cmiUtil::Side side;
cmiUtil::PriceStruct price;
cmiOrder::TimeInForce timeInForce;
cmiUtil::DateTimeStruct expireTime;
cmiOrder::OrderContingencyStruct contingency;
cmiUser::ExchangeFirmStruct cmta;
string extensions;
string account;
string subaccount;

cmiOrder::PositionEffect positionEffect;
cmiOrder::CrossingIndicator cross;
cmiOrder::OriginType orderOriginType;
cmiOrder::Coverage coverage;
string optionalData;
string userAssignedId;
```

```
cmiSession::TradingSessionNameSequence sessionNames;
};
typedef sequence <OrderEntryStruct> OrderEntryStructSequence;

struct LegOrderEntryStruct
{
cmiProduct::ProductKey productKey;
cmiUtil::PriceStruct mustUsePrice;
cmiUser::ExchangeFirmStruct clearingFirm;
cmiOrder::Coverage coverage;
cmiOrder::PositionEffect positionEffect;
};
typedef sequence <LegOrderEntryStruct>
LegOrderEntryStructSequence;
struct LegOrderDetailStruct
{
cmiProduct::ProductKey productKey;
cmiUtil::PriceStruct mustUsePrice;
cmiUser::ExchangeFirmStruct clearingFirm;
cmiOrder::Coverage coverage;
cmiOrder::PositionEffect positionEffect;
cmiUtil::Side side;
long originalQuantity;
long tradedQuantity;
long cancelledQuantity;
long leavesQuantity;
};
typedef sequence <LegOrderDetailStruct>
LegOrderDetailStructSequence;

struct OrderStruct
{
OrderIdStruct orderId;
cmiUser::ExchangeAcronymStruct originator;
// Fields from the OrderEntryStruct
long originalQuantity;
cmiProduct::ProductKey productKey;
cmiUtil::Side side;
cmiUtil::PriceStruct price;
cmiOrder::TimeInForce timeInForce;
cmiUtil::DateTimeStruct expireTime;
cmiOrder::OrderContingencyStruct contingency;
cmiUser::ExchangeFirmStruct cmta;
string extensions;
string account;
string subaccount;

cmiOrder::PositionEffect positionEffect;
cmiOrder::CrossingIndicator cross;
cmiOrder::OriginType orderOriginType;
cmiOrder::Coverage coverage;
string optionalData;
// Additional Order Fields
```

```
string userId;
cmiUser::ExchangeAcronymStruct userAcronym;
cmiProduct::ProductType productType;
cmiProduct::ClassKey classKey;
cmiUtil::DateTimeStruct receivedTime;
cmiOrder::OrderState state;
long tradedQuantity;
long cancelledQuantity;
long leavesQuantity;
cmiUtil::PriceStruct averagePrice;
long sessionTradedQuantity;
long sessionCancelledQuantity;
cmiUtil::PriceStruct sessionAveragePrice;
string orsId;
cmiUtil::Source source;
cmiOrder::OrderIdStruct crossedOrder;
long transactionSequenceNumber;
string userAssignedId;
cmiSession::TradingSessionNameSequence sessionNames;
cmiSession::TradingSessionName activeSession;
cmiOrder::LegOrderDetailStructSequence legOrderDetails;
};
typedef sequence <OrderStruct> OrderStructSequence;
struct OrderDetailStruct
{
cmiProduct::ProductNameStruct productInformation;
cmiUtil::UpdateStatusReason statusChange;
cmiOrder::OrderStruct orderStruct;
};
typedef sequence <OrderDetailStruct>
OrderDetailStructSequence;
struct CancelReportStruct
{
cmiOrder::OrderIdStruct orderId;
cmiUtil::ReportType cancelReportType;
cmiProduct::ProductKey productKey;
cmiSession::TradingSessionName sessionName;
long cancelledQuantity;
long tlcQuantity;
long mismatchedQuantity;
cmiUtil::DateTimeStruct timeSent;
string orsId;
long totalCancelledQuantity;

boolean nothingDone;
long transactionSequenceNumber;
string userAssignedCancelId;
};
typedef sequence <CancelReportStruct>
CancelReportStructSequence;
struct CancelRequestStruct
{
cmiOrder::OrderIdStruct orderId;
cmiSession::TradingSessionName sessionName;
```

```
string userAssignedCancelId;
cmiOrder::CancelType cancelType;
long quantity;
};
struct ContraPartyStruct
{
cmiUser::ExchangeAcronymStruct user;
cmiUser::ExchangeFirmStruct firm;
long quantity;
};
typedef sequence <ContraPartyStruct>
ContraPartyStructSequence;
struct FilledReportStruct
{
cmiUtil::CboeIdStruct tradeId;
cmiUtil::ReportType fillReportType;
cmiUser::ExchangeFirmStruct executingOrGiveUpFirm;
string userId;
cmiUser::ExchangeAcronymStruct userAcronym;
cmiProduct::ProductKey productKey;
cmiSession::TradingSessionName sessionName;
long tradedQuantity;
long leavesQuantity;
cmiUtil::PriceStruct price;
cmiUtil::Side side;
string orsId;
string executingBroker;
cmiUser::ExchangeFirmStruct cmta;
string account;
string subaccount;
cmiUser::ExchangeAcronymStruct originator;
string optionalData;
string userAssignedId;
string extensions;
cmiOrder::ContraPartyStructSequence contraParties;
cmiUtil::DateTimeStruct timeSent;
cmiOrder::PositionEffect positionEffect;
long transactionSequenceNumber;
};
typedef sequence <FilledReportStruct>
FilledReportStructSequence;

struct OrderFilledReportStruct
{
cmiOrder::OrderDetailStruct filledOrder;
cmiOrder::FilledReportStructSequence filledReport;
};
typedef sequence <OrderFilledReportStruct>
OrderFilledReportStructSequence;
struct OrderCancelReportStruct
{
cmiOrder::OrderDetailStruct cancelledOrder;
cmiOrder::CancelReportStructSequence cancelReport;
};
```

```
typedef sequence <OrderCancelReportStruct>
OrderCancelReportStructSequence;
struct PendingOrderStruct {
cmiProduct::PendingNameStruct pendingProductName;
cmiOrder::OrderStruct pendingOrder;
cmiOrder::OrderStruct currentOrder;
};
typedef sequence <PendingOrderStruct>
PendingOrderStructSequence;
struct BustReportStruct
{
cmiUtil::CboeIdStruct tradeId;
cmiUtil::ReportType bustReportType;
cmiSession::TradingSessionName sessionName;
cmiUser::ExchangeFirmStruct executingOrGiveUpFirm;
string userId;
cmiUser::ExchangeAcronymStruct userAcronym;
long bustedQuantity;
cmiUtil::PriceStruct price;
cmiProduct::ProductKey productKey;
cmiUtil::Side side;
cmiUtil::DateTimeStruct timeSent;
long reinstateRequestedQuantity;
long transactionSequenceNumber;
};
typedef sequence <BustReportStruct> BustReportStructSequence;
struct OrderBustReportStruct
{
cmiOrder::OrderDetailStruct bustedOrder;
cmiOrder::BustReportStructSequence bustedReport;
};
typedef sequence <OrderBustReportStruct>
OrderBustReportStructSequence;
struct BustReinstateReportStruct
{
cmiUtil::CboeIdStruct tradeId;

long bustedQuantity;

long reinstatedQuantity;
long totalRemainingQuantity;
cmiUtil::PriceStruct price;
cmiProduct::ProductKey productKey;
cmiSession::TradingSessionName sessionName;
cmiUtil::Side side;
cmiUtil::DateTimeStruct timeSent;
long transactionSequenceNumber;
};
typedef sequence <BustReinstateReportStruct>
BustReinstateReportStructSequence;
struct OrderBustReinstateReportStruct
{
cmiOrder::OrderDetailStruct reinstatedOrder;
cmiOrder::BustReinstateReportStruct bustReinstatedReport;
};
```

```
                typedef sequence <OrderBustReinstateReportStruct>
                OrderBustReinstateReportStructSequence;
                };
```

cmiProduct.idl

```
    module cmiProduct
                {
                typedef char OptionType;
                typedef char ExpirationStyle;
                typedef long ProductKey;
                typedef sequence <ProductKey> ProductKeySequence;
                typedef long ReportingClassKey;
                typedef sequence <ReportingClassKey>
                ReportingClassKeySequence;
                typedef long ClassKey;
                typedef sequence <ClassKey> ClassKeySequence;
                typedef short ProductType;
                typedef sequence <ProductType> ProductTypeSequence;
                typedef string Symbol;
                typedef sequence <Symbol> SymbolSequence;
                typedef double CommodityQuantity;
                typedef short ListingState;
                typedef sequence <ListingState> ListingStateSequence;
                typedef short PriceAdjustmentType;
                typedef short PriceAdjustmentAction;
                typedef short PriceDisplayType;
                typedef string ProductDescriptionName;

                struct ProductDescriptionStruct
                {
                cmiProduct::ProductDescriptionName name;
                cmiProduct::ProductDescriptionName baseDescriptionName;
                cmiUtil::PriceStruct minimumStrikePriceFraction;
                cmiUtil::PriceStruct maxStrikePrice;
                cmiUtil::PriceStruct premiumBreakPoint;
                cmiUtil::PriceStruct minimumAbovePremiumFraction;
                cmiUtil::PriceStruct minimumBelowPremiumFraction;
                cmiProduct::PriceDisplayType priceDisplayType;
                cmiProduct::PriceDisplayType premiumPriceFormat;
                cmiProduct::PriceDisplayType strikePriceFormat;
                cmiProduct::PriceDisplayType underlyingPriceFormat;
                };
                typedef sequence <ProductDescriptionStruct>
                ProductDescriptionStructSequence;

                struct ProductNameStruct
                {
                cmiProduct::Symbol reportingClass;
                cmiUtil::PriceStruct exercisePrice;
                cmiUtil::DateStruct expirationDate;
                cmiProduct::OptionType optionType;
                cmiProduct::Symbol productSymbol;
                };
```

```
typedef sequence <ProductNameStruct>
ProductNameStructSequence;

struct ProductKeysStruct {
cmiProduct::ProductKey productKey;
cmiProduct::ClassKey classKey;
cmiProduct::ProductType productType;
cmiProduct::ReportingClassKey reportingClass;
};
struct ListingStateStruct
{
cmiProduct::ProductKeysStruct productKeys;
cmiProduct::ListingState productState;
};
typedef sequence <ListingStateStruct>
ListingStateStructSequence;
struct ProductStruct
{
cmiProduct::ProductKeysStruct productKeys;
cmiProduct::ProductNameStruct productName;
cmiProduct::ListingState listingState;
string description;
string companyName;
string unitMeasure;
cmiProduct::CommodityQuantity standardQuantity;
cmiUtil::DateStruct maturityDate;
cmiUtil::DateStruct activationDate;
cmiUtil::DateStruct inactivationDate;
cmiUtil::DateTimeStruct createdTime;
cmiUtil::DateTimeStruct lastModifiedTime;
char opraMonthCode;
char opraPriceCode;
};
typedef sequence <ProductStruct> ProductStructSequence;
struct EPWStruct {
double minimumBidRange;
double maximumBidRange;
double maximumAllowableSpread;
};
typedef sequence <EPWStruct> EPWStructSequence;
struct ClassStruct
{
cmiProduct::ClassKey classKey;
cmiProduct::ProductType productType;
cmiProduct::ListingState listingState;
cmiProduct::Symbol classSymbol;
cmiProduct::ProductStruct underlyingProduct;
cmiProduct::Symbol primaryExchange;
cmiUtil::DateStruct activationDate;
cmiUtil::DateStruct inactivationDate;
cmiUtil::DateTimeStruct createdTime;
cmiUtil::DateTimeStruct lastModifiedTime;
cmiProduct::EPWStructSequence epwValues;

double epwFastMarketMultiplier;
```

```
cmiProduct::ProductDescriptionStruct productDescription;
boolean testClass;
};
typedef sequence <ClassStruct> ClassStructSequence;
struct ProductTypeStruct {
cmiProduct::ProductType type;
string name;
string description;
cmiUtil::DateTimeStruct createdTime;
cmiUtil::DateTimeStruct lastModifiedTime;
};
typedef sequence <ProductTypeStruct>
ProductTypeStructSequence;

struct PendingNameStruct
{
cmiProduct::PriceAdjustmentAction action;
cmiProduct::ProductStruct productStruct;
cmiProduct::ProductNameStruct pendingProductName;
};
typedef sequence <PendingNameStruct>
PendingNameStructSequence;

struct PendingAdjustmentStruct
{
cmiProduct::ClassKey classKey;
cmiUtil::DateStruct effectiveDate;
cmiUtil::DateStruct submittedDate;
cmiProduct::PriceAdjustmentType type;
boolean active;
cmiProduct::PendingNameStructSequence productsPending;
};
typedef sequence <PendingAdjustmentStruct>
PendingAdjustmentStructSequence;
};
```

cmiQuote.idl

```
module cmiQuote
            {
            typedef short RFQType;
            typedef long QuoteKey;
            typedef sequence <QuoteKey> QuoteKeySequence;
            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 QuoteStruct
{
cmiQuote::QuoteKey quoteKey;
cmiProduct::ProductKey productKey;
cmiSession::TradingSessionName sessionName;
string userId;
string clearingFirm;
cmiUtil::PriceStruct bidPrice;
long bidQuantity;
cmiUtil::PriceStruct askPrice;
long askQuantity;
long transactionSequenceNumber;
string userAssignedId;
};
typedef sequence <QuoteStruct> QuoteStructSequence;
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 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;
};
```

cmiSession.idl

```
module cmiSession
{
typedef short TradingSessionState;
typedef short LoginSessionType;
typedef string TradingSessionName;
typedef sequence <TradingSessionName>
TradingSessionNameSequence;
typedef short ProductState;
typedef sequence <ProductState> ProductStateSequence;
typedef short ClassState;
typedef sequence <ClassState> ClassStateSequence;
struct TradingSessionStateStruct
{
cmiSession::TradingSessionName sessionName;
```

```
cmiSession::TradingSessionState sessionState;
};
typedef sequence <TradingSessionStateStruct>
TradingSessionStateStructSequence;
struct TradingSessionStruct
{
cmiSession::TradingSessionName sessionName;
cmiSession::TradingSessionState state;
cmiUtil::TimeStruct startTime;
cmiUtil::TimeStruct endTime;
long sequenceNumber;
};
typedef sequence <TradingSessionStruct>
TradingSessionStructSequence;
struct ProductStateStruct
{
cmiProduct::ProductKeysStruct productKeys;
cmiSession::TradingSessionName sessionName;
cmiSession::ProductState productState;
long productStateTransactionSequenceNumber;
};
typedef sequence <ProductStateStruct>
ProductStateStructSequence;
struct ClassStateStruct
{
cmiProduct::ClassKey classKey;
cmiSession::TradingSessionName sessionName;
cmiSession::ClassState classState;
long classStateTransactionSequenceNumber;
};
typedef sequence <ClassStateStruct> ClassStateStructSequence;
struct SessionClassStruct
{

cmiSession::TradingSessionName sessionName;
cmiSession::TradingSessionName underlyingSessionName;
cmiSession::TradingSessionNameSequence eligibleSessions;
cmiSession::ClassState classState;
cmiProduct::ClassStruct classStruct;
long classStateTransactionSequenceNumber;
};
typedef sequence <SessionClassStruct>
SessionClassStructSequence;
struct SessionProductStruct
{
cmiSession::TradingSessionName sessionName;
cmiSession::ProductState productState;
cmiProduct::ProductStruct productStruct;
long productStateTransactionSequenceNumber;
};
typedef sequence <SessionProductStruct>
SessionProductStructSequence;
struct SessionStrategyLegStruct
{
cmiSession::TradingSessionName sessionName;
```

```
cmiProduct::ProductKey product;
long ratioQuantity;
cmiUtil::Side side;
};
typedef sequence <SessionStrategyLegStruct>
SessionStrategyLegStructSequence;
struct SessionStrategyStruct
{
cmiStrategy::StrategyType strategyType;
cmiSession::SessionProductStruct sessionProductStruct;
cmiSession::SessionStrategyLegStructSequence
sessionStrategyLegs;
};
typedef sequence <SessionStrategyStruct>
SessionStrategyStructSequence;
struct SessionClassDetailStruct
{
cmiSession::SessionClassStruct classDetail;
cmiSession::SessionProductStructSequence products;
};
typedef sequence <SessionClassDetailStruct>
SessionClassDetailStructSequence;
};
```

cmiStrategy.idl

```
module cmiStrategy
{
typedef short StrategyType;
typedef short MonthIncrement;
struct StrategyLegStruct
{
cmiProduct::ProductKey product;
long ratioQuantity;
cmiUtil::Side side;
};
typedef sequence <StrategyLegStruct>
StrategyLegStructSequence;
struct StrategyStruct
{
cmiProduct::ProductStruct product;
cmiStrategy::StrategyType strategyType;
StrategyLegStructSequence strategyLegs;
};
typedef sequence <StrategyStruct> StrategyStructSequence;
struct StrategyRequestStruct
{
cmiStrategy::StrategyLegStructSequence strategyLegs;
};
typedef sequence <StrategyRequestStruct>
StrategyRequestStructSequence;
};
```

cmiTraderActivity.idl

No changes between versions 1.0b and 2.0.

cmiUser.idl

```
module cmiUser
{
typedef char UserRole;
typedef char LoginSessionMode;
struct ExchangeFirmStruct
{
string exchange;
string firmNumber;
};
typedef sequence <ExchangeFirmStruct>
ExchangeFirmStructSequence;
struct ExchangeAcronymStruct
{
string 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 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;
                };

        };
```

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 QueryDirection;
                typedef sequence <string> StringSequence;
                typedef sequence <long> LongSequence;
                typedef double PricingModelParameter;
                typedef short ReportType;
                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;
                };
        };
```

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 LoginSessionTypes
                {
                const cmiSession::LoginSessionType PRIMARY = 1;
                const cmiSession::LoginSessionType SECONDARY = 2;
                };
                interface MarketDataHistoryEntryTypes
                {
                const cmiMarketData::MarketDataHistoryEntryType
                QUOTE_ENTRY = 1;
                const cmiMarketData::MarketDataHistoryEntryType
                PRICE_REPORT_ENTRY = 2;
                const cmiMarketData::MarketDataHistoryEntryType
                EXPECTED_OPEN_PRICE = 3;
                const cmiMarketData::MarketDataHistoryEntryType
                MARKET_CONDITION_ENTRY = 4;
                const cmiMarketData::MarketDataHistoryEntryType
```

```
UNSIZED_QUOTE_ENTRY = 5;
};
interface MarketChangeReasons
{
const cmiMarketData::MarketChangeReason EXCHANGE = 1;
const cmiMarketData::MarketChangeReason NBBO = 2;
const cmiMarketData::MarketChangeReason COMBINED = 3;
};
interface QueryDirections
{
const cmiUtil::QueryDirection QUERY_FORWARD = 1;
const cmiUtil::QueryDirection QUERY_BACKWARD = 2;
};
interface PriceDisplayTypes
{
const cmiProduct::PriceDisplayType FRACTION = 1;
const cmiProduct::PriceDisplayType DECIMAL = 2;
};
interface OptionTypes
{
const cmiProduct::OptionType CALL = 'C';

const cmiProduct::OptionType PUT = 'P';
};
interface ProductTypes
{
const cmiProduct::ProductType COMMODITY = 1;
const cmiProduct::ProductType DEBT = 2;
const cmiProduct::ProductType EQUITY = 3;
const cmiProduct::ProductType FUTURE = 4;
const cmiProduct::ProductType INDEX = 5;
const cmiProduct::ProductType LINKED_NOTE = 6;
const cmiProduct::ProductType OPTION = 7;
const cmiProduct::ProductType UNIT_INVESTMENT_TRUST = 8;
const cmiProduct::ProductType VOLATILITY_INDEX = 9;
const cmiProduct::ProductType WARRANT = 10;
const cmiProduct::ProductType STRATEGY = 11;
};
interface ProductStates
{
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;
};
interface ListingStates
{
const cmiProduct::ListingState ACTIVE = 1;
const cmiProduct::ListingState INACTIVE = 2;
const cmiProduct::ListingState UNLISTED = 3;
```

```
const cmiProduct::ListingState OBSOLETE = 4;
};
interface ClassStates
{
const cmiSession::ClassState NOT_IMPLEMENTED = 1;
};
interface TradingSessionStates
{
const cmiSession::TradingSessionState CLOSED = 1;
const cmiSession::TradingSessionState OPEN = 2;
};
interface TradingSessionType
{
const cmiSession::TradingSessionState DAY = 1;
const cmiSession::TradingSessionState EVENING = 2;
};

interface TradingSessionMethod
{
const cmiSession::TradingSessionState SBT = 1;
const cmiSession::TradingSessionState OPENOUTCRY = 2;
};
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 REMOVED = 11;
};
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
```

```
};
interface VolumeTypes
{
const cmiMarketData::VolumeType LIMIT = 1; // Limit (no
contingency)
const cmiMarketData::VolumeType AON = 2; // All or None
const cmiMarketData::VolumeType FOK = 3; // Fill or Kill
const cmiMarketData::VolumeType IOC = 4; // Immediate or
Cancel
};
interface OrderStates
{

const cmiOrder::OrderState BOOKED = 1;
const cmiOrder::OrderState CANCEL = 2;
const cmiOrder::OrderState FILL = 3;
const cmiOrder::OrderState OPEN_OUTCRY = 4;
const cmiOrder::OrderState INACTIVE = 5;
const cmiOrder::OrderState ACTIVE = 6;
const cmiOrder::OrderState EXPIRED = 7;
const cmiOrder::OrderState PURGED = 8;
const cmiOrder::OrderState REMOVED = 9;
};
interface RFQTypes
{
const cmiQuote::RFQType MANUAL = 1;
const cmiQuote::RFQType SYSTEM = 2;
};
interface Sides
{
const cmiUtil::Side BUY = 'B';
const cmiUtil::Side SELL = 'S';
const cmiUtil::Side BID = 'B';
const cmiUtil::Side ASK = 'A';
const cmiUtil::Side AS_DEFINED = 'D';
const cmiUtil::Side OPPOSITE = 'O';
};
interface TimesInForce
{
const cmiOrder::TimeInForce GTC = 'G';
const cmiOrder::TimeInForce DAY = 'D';
const cmiOrder::TimeInForce GTD = 'T'; // Good until
datetime
};
interface PositionEffects
{
const cmiOrder::PositionEffect OPEN = 'O';
const cmiOrder::PositionEffect CLOSED = 'C';
const cmiOrder::PositionEffect NOTAPPLICABLE = 'N';
};
interface OrderOrigins
{
const cmiOrder::OriginType CUSTOMER = 'C';
const cmiOrder::OriginType FIRM = 'F';
const cmiOrder::OriginType BROKER_DEALER = 'B';
```

```
const cmiOrder::OriginType CUSTOMER_BROKER_DEALER = 'X';
const cmiOrder::OriginType MARKET_MAKER = 'M';
};

interface UserRoles
{
const cmiUser::UserRole FIRM = 'F';
const cmiUser::UserRole BROKER_DEALER = 'B';

const cmiUser::UserRole CUSTOMER_BROKER_DEALER = 'X';
const cmiUser::UserRole MARKET_MAKER = 'M';
const cmiUser::UserRole HELP_DESK = 'H';
const cmiUser::UserRole DPM_ROLE = 'D';
const cmiUser::UserRole UNKNOWN_ROLE = 'K';
const cmiUser::UserRole CLASS_DISPLAY = 'C';
const cmiUser::UserRole FIRM_DISPLAY = 'R';
};
interface EntryTypes
{
const cmiUtil::EntryType ADD = 'A';
const cmiUtil::EntryType CANCEL = 'C';
const cmiUtil::EntryType CANCEL_REPLACE = 'R';
const cmiUtil::EntryType FILL = 'F';
const cmiUtil::EntryType BOOK = 'B';
const cmiUtil::EntryType STATE = 'S';
const cmiUtil::EntryType PRICE_ADJUST = 'P';
const cmiUtil::EntryType UPDATE = 'U';
const cmiUtil::EntryType BUST = 'K';
};
interface CoverageTypes
{
const cmiOrder::Coverage UNSPECIFIED = 'B';
const cmiOrder::Coverage COVERED = 'C';
const cmiOrder::Coverage UNCOVERED = 'U';
};
interface Sources
{
const cmiUtil::Source TPF = 'T';
const cmiUtil::Source SBT = 'S';
};
interface PriceTypes
{
const cmiUtil::PriceType NO_PRICE = 1;
const cmiUtil::PriceType LIMIT = 2;
const cmiUtil::PriceType VALUED = 2;
const cmiUtil::PriceType MARKET = 3;
};
interface ExpirationStyles
{
const cmiProduct::ExpirationStyle EUROPEAN = 'E';
const cmiProduct::ExpirationStyle AMERICAN = 'A';
};
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;
};
interface PriceAdjustmentTypes
{
const cmiProduct::PriceAdjustmentType SPLIT = 1;
const cmiProduct::PriceAdjustmentType DIVIDEND_CASH = 2;
const cmiProduct::PriceAdjustmentType DIVIDEND_PERCENT =
3;
const cmiProduct::PriceAdjustmentType DIVIDEND_STOCK = 4;
const cmiProduct::PriceAdjustmentType LEAP_ROLLOVER = 5;
const cmiProduct::PriceAdjustmentType MERGER = 6;
const cmiProduct::PriceAdjustmentType SYMBOL_CHANGE = 7;
const cmiProduct::PriceAdjustmentType COMMON_DISTRIBUTION
=8;
};
interface PriceAdjustmentActions
{
const cmiProduct::PriceAdjustmentAction
PRICE_ADJUSTMENT_UPDATE = 1;
const cmiProduct::PriceAdjustmentAction
PRICE_ADJUSTMENT_DELETE = 2;
const cmiProduct::PriceAdjustmentAction
PRICE_ADJUSTMENT_CREATE = 3;
const cmiProduct::PriceAdjustmentAction
PRICE_ADJUSTMENT_MOVE = 4;
};
interface PriceScale
{
const long DEFAULT_SCALE = 1000000000;
};
interface ProductClass
{
const long DEFAULT_CLASS_KEY = 0;
};
interface OrderCancelTypes
{
const cmiOrder::CancelType DESIRED_CANCEL_QUANTITY =
1;
const cmiOrder::CancelType DESIRED_REMAINING_QUANTITY =
2;
const cmiOrder::CancelType CANCEL_ALL_QUANTITY =
```

```
3;
};
interface ActivityTypes
{
// Order Activity Events
const cmiTraderActivity::ActivityType NEW_ORDER
=1;
const cmiTraderActivity::ActivityType FILL_ORDER
=2;
const cmiTraderActivity::ActivityType CANCEL_ORDER
=3;
const cmiTraderActivity::ActivityType BUST_ORDER_FILL
=4;
const cmiTraderActivity::ActivityType
BUST_REINSTATE_ORDER = 5;
const cmiTraderActivity::ActivityType
CANCEL_REPLACE_ORDER = 6;
const cmiTraderActivity::ActivityType UPDATE_ORDER
=7;
const cmiTraderActivity::ActivityType BOOK_ORDER
=8;
const cmiTraderActivity::ActivityType STATE_CHANGE_ORDER
=9;
const cmiTraderActivity::ActivityType PRICE_ADJUST_ORDER
= 10;
const cmiTraderActivity::ActivityType CANCEL_ALL_ORDERS
= 11;
// Strategy Order leg activity types
const cmiTraderActivity::ActivityType
NEW_ORDER_STRATEGY_LEG = 51;
const cmiTraderActivity::ActivityType FILL_STRATEGY_LEG
= 52;
const cmiTraderActivity::ActivityType
CANCEL_STRATEGY_LEG = 53;
const cmiTraderActivity::ActivityType
BUST_STRATEGY_LEG_FILL = 54;
const cmiTraderActivity::ActivityType
BUST_REINSTATE_STRATEGY_LEG = 55;
const cmiTraderActivity::ActivityType
UPDATE_STRATEGY_LEG = 57;
const cmiTraderActivity::ActivityType
PRICE_ADJUST_ORDER_LEG = 60;
// Quote Activity Events
const cmiTraderActivity::ActivityType NEW_QUOTE
= 101;
const cmiTraderActivity::ActivityType FILL_QUOTE
= 102;
const cmiTraderActivity::ActivityType CANCEL_QUOTE
= 103;
const cmiTraderActivity::ActivityType CANCEL_ALL_QUOTES
= 104;
const cmiTraderActivity::ActivityType
SYSTEM_CANCEL_QUOTE = 105;
const cmiTraderActivity::ActivityType UPDATE_QUOTE
= 106;
```

```
const cmiTraderActivity::ActivityType BUST_QUOTE_FILL
= 107;
// Strategy Quote leg activity types
const cmiTraderActivity::ActivityType QUOTE_LEG_FILL
= 152;
const cmiTraderActivity::ActivityType
BUST_QUOTE_LEG_FILL = 157;
// RFQ Activity Events
const cmiTraderActivity::ActivityType NEW_RFQ
= 201;
};
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;
};
interface ActivityReasons
{
const cmiTraderActivity::ActivityReason
NOTHING_DONE = 1; // used with CANCEL_REASON
const cmiTraderActivity::ActivityReason USER
= 2; // used with CANCEL_REASON
const cmiTraderActivity::ActivityReason SYSTEM
= 3; // used with CANCEL_REASON
};
```

```
interface StrategyTypes
{
const cmiStrategy::StrategyType UNKNOWN = 1;
const cmiStrategy::StrategyType STRADDLE = 2;
const cmiStrategy::StrategyType PSEUDO_STRADDLE = 3;
const cmiStrategy::StrategyType VERTICAL = 4;
const cmiStrategy::StrategyType RATIO = 5;
const cmiStrategy::StrategyType TIME = 6;
const cmiStrategy::StrategyType DIAGONAL = 7;
const cmiStrategy::StrategyType COMBO = 8;
};
interface ReportTypes
{
const cmiUtil::ReportType REGULAR_REPORT = 1;
const cmiUtil::ReportType STRATEGY_REPORT = 2;
const cmiUtil::ReportType STRATEGY_LEG_REPORT = 3;
```

```
        };
        interface BookDepthTypes
        {
        const cmiMarketData::BookDepthUpdateType DELETE_PRICE =
        'D'; // delete book price
        const cmiMarketData::BookDepthUpdateType INSERT_PRICE =
        'I'; // new book price
        const cmiMarketData::BookDepthUpdateType UPDATE_PRICE =
        'U'; // book price update
        };
};
```

cmiConstants.idlExample Program Changes

Java Examples

Common

No change from beta 2 to Production.

Example 1

Displays the output for Expiration date for Futures products

Example 2

Added a new method, acceptQuoteCancelReport, to the QuoteStatusConsumerCallback

Example 3

Replaced CancelReasons with AcitvityReasons constants

Example 4

No change from beta 2 to Production.

Example 5

No change from beta 2 to Production.

Example 6

No change from beta 2 to Production.

C++ Examples

Refer to the Java examples for changes.

Documentation Changes

API-01

Updated to reflect changes in CMi.

API-02

Updated to reflect changes in CMi.

API-03

Updated to reflect changes in CMi.

API-04

Updated to reflect changes in CMi.

API-05

Updated to reflect changes in CMi.

API-06

Updated to reflect changes in CMi.

API-07

Updated to reflect changes in CMi.

API-08

Obsolete.

Simulator Changes

Added additional Futures products.

New spread type BUY_WRITE

Spreads can be created on Future products only for AXP and AIG classes