CONFIDENTIAL

CHICAGO BOARD OPTIONS EXCHANGE

CBOE Application Programming Interface

CBOE API Version 2.1 - Release Notes

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

CBOE PROPRIETARY INFORMATION

24 January 2003

Document #[API-00]

Production Release - Update

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

Introduction

This document describes the changes that have been introduced to the CMi API. IDL changes are detailed below for the release of the CMi Version 2.1.

Note: New interfaces for the NBBO IDL will be reflected in the next release of the CMi.

CMi Release Schedule

Version	Description	Planned Date
2.1	Production Release - Update	January 2003
2.0.1	Production Release - Update	May 2002
2.0	Production Release	April 2002

Summary of Changes to the CMi IDL Version 2.1 - Production Release Update

This section describes the changes in the IDL between the CMi Version 2.1 and CMi Version 2.0.1.

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

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

Added new origin types for Linkage

- PRINCIPAL
- PRINCIPAL_ACTING_AS_AGENT
- SATISFACTION

Added a new user role: TFL_ROLE

Added a new source for Linkage

Added activity types for:

- Linkage:
    - NEW_ORDER_REJECT
    - FILL_REJECT
    - CANCEL_ORDER_REQUEST
    - CANCEL_ORDER_REQUEST_REJECT
    - CANCEL_REPORT_REJECT
    - NEW_ORDER_REJECT_REJECTED
    - FILL_REJECT_REJECTED
    - CANCEL_ORDER_REQUEST_REJECT_REJECTED
    - CANCEL_REPORT_REJECT_REJECTED
- HELD_FOR_IPP_PROTECTION
- CANCEL_REPLACE_ORDER_REQUEST

Added new activity reasons for:

- Linkage:
    - BROKER_OPTION
    - CANCEL_PENDING
    - CROWD_TRADE
    - DUPLICATE_ORDER
    - EXCHANGE_CLOSED
    - GATE_VIOLATION
    - INVALID_ACCOUNT
    - INVALID_AUTOEX_VALUE

- INVALID_CMTA
- INVALID_FIRM
- INVALID_ORIGIN_TYPE
- INVALID_POSITION_EFFECT
- INVALID_PRICE
- INVALID_PRODUCT
- INVALID_PRODUCT_TYPE
- INVALID_QUANITY
- INVALID_SIDE
- INVALID_SUBACCOUNT
- INVALID_TIME_IN_FORCE
- INVALID_USER
- LATE_PRINT
- NOT_FIRM
- MISSING_EXEC_INFO
- NO_MATCHING_ORDER
- NON_BLOCK_TRADE
- NOT_NBBO
- COMM_DELAYS
- ORIGINAL_ORDER_REJECTED
- OTHER
- PROCESSING_PROBLEMS
- PRODUCT_HALTED
- PRODUCT_IN_ROTATION
- STALE_EXECUTION
- STALE_ORDER
- ORDER_TOO_LATE
- TRADE_BUSTED
- TRADE_REJECTED
- UNKNOWN_ORDER
- INVALD_EXCHANGE
- TRANSACTION_FAILED
- NOT_ACCEPTED
- AWAY_EXCHANGE_CANCEL

  - LINKAGE_CONDITIONAL_FIELD_MISSING
  - LINKAGE_EXCHANGE_UNAVAILABLE
  - LINKAGE_INVALID_MESSAGE
  - LINKAGE_INVALID_DESTINATION
  - LINKAGE_INVALID_PRODUCT
  - LINKAGE_SESSION_REJECT
- INSUFFICIENT_QUANTITY_BUY_SIDE
- INSUFFICIENT_QUANTITY_SELL_SIDE

New report types:

- NEW_ORDER_REJECT
- FILL_REJECT
- CANCEL_ORDER_REQUEST
- CANCEL_ORDER_REQUEST_REPORT
- CANCEL_REPORT_REJECT
- NEW_ORDER_REJECT_REJECTED
- FILL_REJECT_REJECTED
- CANCEL_ORDER_REQUEST_REJECT_REJECTED
- CANCEL_REPORT_REJECT_REJECTED

Added new interfaces:

- Interface HandlingInstructions
- Interface ExtensionFields
- Interface ExchangeMarketInfoType

cmiIntermarket.idl

New module in the CMi.

cmiIntermarketCallback.idl

New module in the CMi.

cmiIntermarketMessages.idl

New module in the CMi.

cmiErrorCodes.idl

Added an INVALID_EXCHANGE error code for Linkage support.

IDL Changes for Version 2.1

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

No change between the CMi Version 2.1 and Version 2.0.1.

Quote Interface

No change between the CMi Version 2.1 and Version 2.0.1.

OrderQuery Interface

No change between the CMi Version 2.1 and Version 2.0.1.

ProductQuery Interface

No change between the CMi Version 2.1 and Version 2.0.1.

ProductDefinition Interface

No change between the CMi Version 2.1 and Version 2.0.1.

OrderEntry Interface

No change between the CMi Version 2.1 and Version 2.0.1.

UserPreferenceQuery Interface

No change between the CMi Version 2.1 and Version 2.0.1.

Administrator Interface

No change between the CMi Version 2.1 and Version 2.0.1.

TradingSession Interface

No change between the CMi Version 2.1 and Version 2.0.1.

UserHistory Interface

No change between the CMi Version 2.1 and Version 2.0.1.

UserTradingParameters Interface

No change between the CMi Version 2.1 and Version 2.0.1.

UserSessionManager Interface

No change between the CMi Version 2.1 and Version 2.0.1.

UserAccess Interface

No change between the CMi Version 2.1 and Version 2.0.1.

cmiAdmin.idl

No change between the CMi Version 2.1 and Version 2.0.1.

cmiCallback.idl

No change between the CMi Version 2.1 and Version 2.0.1.

cmiMarketData.idl

No change between the CMi Version 2.1 and Version 2.0.1.

cmiOrder.idl

No change between the CMi Version 2.1 and Version 2.0.1.

cmiProduct.idl

No change between the CMi Version 2.1 and Version 2.0.1.

cmiQuote.idl

No change between the CMi Version 2.1 and Version 2.0.1.

cmiSession.idl

No change between the CMi Version 2.1 and Version 2.0.1.

cmiStrategy.idl

No change between the CMi Version 2.1 and Version 2.0.1.

cmiTraderActivity.idl

No change between the CMi Version 2.1 and Version 2.0.1.

cmiUser.idl

No change between the CMi Version 2.1 and Version 2.0.1.

cmiUtil.idl

No change between the CMi Version 2.1 and Version 2.0.1.

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;
        };

    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';          //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 MARKET_MAKER_AWAY = 'N';
              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

        // Special "MARKET MAKER" origin types for Linkage
        const cmiOrder::OriginType PRINCIPAL = 'P';
        const cmiOrder::OriginType PRINCIPAL_ACTING_AS_AGENT = 'A';
        const cmiOrder::OriginType SATISFACTION = 'S';
    };

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';
        const cmiUser::UserRole TFL_ROLE = 'T';
    };

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';
        const cmiUtil::Source LINKAGE = 'L';
    };

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;
            const cmiTraderActivity::ActivityType  HELD_FOR_IPP_PROTECTION = 12;
// new for IPP
            const cmiTraderActivity::ActivityType  CANCEL_REPLACE_ORDER_REQUEST =
13;

            // 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;

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

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;
            const cmiUtil::ActivityReason INSUFFICIENT_QUANTITY_BUY_SIDE  = 8;
            const cmiUtil::ActivityReason INSUFFICIENT_QUANTITY_SELL_SIDE  = 9;

            // The following are used for Linkage

            const cmiUtil::ActivityReason BROKER_OPTION = 100;
            const cmiUtil::ActivityReason CANCEL_PENDING = 101;
            const cmiUtil::ActivityReason CROWD_TRADE = 102;
            const cmiUtil::ActivityReason DUPLICATE_ORDER = 103;
            const cmiUtil::ActivityReason EXCHANGE_CLOSED = 104;
            const cmiUtil::ActivityReason GATE_VIOLATION = 105;
            const cmiUtil::ActivityReason INVALID_ACCOUNT = 106;
            const cmiUtil::ActivityReason INVALID_AUTOEX_VALUE = 107;
            const cmiUtil::ActivityReason INVALID_CMTA = 108;
            const cmiUtil::ActivityReason INVALID_FIRM = 109;
            const cmiUtil::ActivityReason INVALID_ORIGIN_TYPE = 110;
            const cmiUtil::ActivityReason INVALID_POSITION_EFFECT = 111;
            const cmiUtil::ActivityReason INVALID_PRICE = 112;
            const cmiUtil::ActivityReason INVALID_PRODUCT = 113;
            const cmiUtil::ActivityReason INVALID_PRODUCT_TYPE = 114;
            const cmiUtil::ActivityReason INVALID_QUANTITY = 115;
            const cmiUtil::ActivityReason INVALID_SIDE = 116;
            const cmiUtil::ActivityReason INVALID_SUBACCOUNT = 117;
            const cmiUtil::ActivityReason INVALID_TIME_IN_FORCE = 118;
            const cmiUtil::ActivityReason INVALID_USER = 119;
            const cmiUtil::ActivityReason LATE_PRINT = 120;
            const cmiUtil::ActivityReason NOT_FIRM = 121;
            const cmiUtil::ActivityReason MISSING_EXEC_INFO = 122;
            const cmiUtil::ActivityReason NO_MATCHING_ORDER = 123;
            const cmiUtil::ActivityReason NON_BLOCK_TRADE = 124;
            const cmiUtil::ActivityReason NOT_NBBO = 125;
            const cmiUtil::ActivityReason COMM_DELAYS = 126;
```

```
        const cmiUtil::ActivityReason ORIGINAL_ORDER_REJECTED = 127;
        const cmiUtil::ActivityReason OTHER = 128;
        const cmiUtil::ActivityReason PROCESSING_PROBLEMS = 129;
        const cmiUtil::ActivityReason PRODUCT_HALTED = 130;
        const cmiUtil::ActivityReason PRODUCT_IN_ROTATION = 131;
        const cmiUtil::ActivityReason STALE_EXECUTION = 132;
        const cmiUtil::ActivityReason STALE_ORDER = 133;
        const cmiUtil::ActivityReason ORDER_TOO_LATE = 134;
        const cmiUtil::ActivityReason TRADE_BUSTED = 135;
        const cmiUtil::ActivityReason TRADE_REJECTED = 136;

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

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

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

    };

interface 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;

        const cmiUtil::ReportType NEW_ORDER_REJECT = 101;
        const cmiUtil::ReportType FILL_REJECT = 102;
```

```
//        const cmiUtil::ReportType CANCEL_ORDER_REQUEST = 103; // not needed
- not used in CancelReportStruct
        const cmiUtil::ReportType CANCEL_ORDER_REQUEST_REJECT = 104;
        const cmiUtil::ReportType CANCEL_REPORT_REJECT = 105;
        const cmiUtil::ReportType NEW_ORDER_REJECT_REJECTED = 106;
        const cmiUtil::ReportType FILL_REJECT_REJECTED = 107;
        const cmiUtil::ReportType CANCEL_ORDER_REQUEST_REJECT_REJECTED = 108;
        const cmiUtil::ReportType CANCEL_REPORT_REJECT_REJECTED = 109;
    };

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;
    };

    interface HandlingInstructions
    {
        const cmiIntermarketMessages::HandlingInstruction AUTOMATIC = 1;
        const cmiIntermarketMessages::HandlingInstruction MANUAL = 2;
        const cmiIntermarketMessages::HandlingInstruction OTHER = 3;
    };


    typedef string ExtensionField;


    interface ExtensionFields
    {

        // Used for routing an order to a BART terminal

        const ExtensionField BARTID = "BARTID";

        // Firm information for stock leg of a buy-write

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

        // The following are used for linkage

        const ExtensionField CBOE_EXEC_ID = "cboeExecId";

        const ExtensionField AWAY_EXCHANGE_USER_ACRONYM="1";
        const ExtensionField USER_ASSIGNED_CANCEL_ID="11";
```

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

    interface ExchangeMarketInfoType
    {
        const cmiIntermarketMessages::ExchangeMarketInfoType
NBBO_ORDER_RECEIVED = 1;
        const cmiIntermarketMessages::ExchangeMarketInfoType
NBBO_ORDER_EXECUTED = 2;
        const cmiIntermarketMessages::ExchangeMarketInfoType
BBO_ORDER_RECEIVED = 3;
        const cmiIntermarketMessages::ExchangeMarketInfoType
BBO_ORDER_EXECUTED = 4;
        const cmiIntermarketMessages::ExchangeMarketInfoType
WORST_NBBO_IN_TIME_WINDOW = 5;
        const cmiIntermarketMessages::ExchangeMarketInfoType
WORST_BBO_IN_TIME_WINDOW = 6;
    };


};
```

cmiIntermarket.idl

```
module cmiIntermarket {

    interface IntermarketQuery
    {
        cmiIntermarketMessages::CurrentIntermarketStruct
getIntermarketByProductForSession(
            in cmiProduct::ProductKey productKey,
            in cmiSession::TradingSessionName session )
                raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
```

```
                    exceptions::DataValidationException
                    exceptions::NotAcceptedException
                );
        cmiIntermarketMessages::CurrentIntermarketStructSequence
getIntermarketByClassForSession(
            in cmiProduct::ClassKey classKey,
            in cmiSession::TradingSessionName session )
                raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::DataValidationException,
                    exceptions::NotAcceptedException
                );
    };

    interface IntermarketHeldOrderEntry
    {
        void rerouteHeldOrder(
            in cmiOrder::OrderIdStruct heldOrderId,
            in cmiSession::TradingSessionName session,
            in cmiProduct::ProductKey productKey,
            in boolean nbboProtectionFlag)
                raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::DataValidationException,
                    exceptions::TransactionFailedException,
                    exceptions::NotAcceptedException

        void rerouteHeldOrderByClass(
            in cmiProduct::ClassKey classKey,
            in cmiSession::TradingSessionName session,
            in boolean nbboProtectionFlag)
                raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::DataValidationException,
                    exceptions::TransactionFailedException,
                    exceptions::NotAcceptedException
        );

        void acceptCancelResponse(
            in cmiOrder::OrderIdStruct orderId,
            in cmiUtil::CboeIdStruct cancelRequestId,
            in cmiSession::TradingSessionName session,
            in cmiProduct::ProductKey productKey)
                raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::NotAcceptedException,
```

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

        void acceptFillHeldOrder(
            in cmiOrder::OrderIdStruct heldOrderId,
            in cmiSession::TradingSessionName session,
            in cmiOrder::OrderEntryStruct nbboAgentOrder)
                raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::NotAcceptedException,
                    exceptions::TransactionFailedException,
                    exceptions::DataValidationException
        );
     };
    interface IntermarketHeldOrderQuery {

        cmiIntermarketMessages::HeldOrderDetailStruct getHeldOrderById(
          in cmiSession::TradingSessionName session,
          in cmiProduct::ProductKey productKey,
          in cmiOrder::OrderIdStruct orderId)
                raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::DataValidationException,
                    exceptions::NotFoundException
        );
     };

    interface NBBOAgentSessionManager
    {
        IntermarketHeldOrderEntry getIntermarketHeldOrderEntry()
            raises(
                exceptions::SystemException,
                exceptions::CommunicationException,
                exceptions::AuthorizationException
            );

        IntermarketHeldOrderQuery getIntermarketHeldOrderQuery()
            raises(
                exceptions::SystemException,
                exceptions::CommunicationException,
                exceptions::AuthorizationException
        );
     };

    interface NBBOAgent
    {
        NBBOAgentSessionManager registerAgent(
            in cmiProduct::ClassKey classKey,
            in cmiSession::TradingSessionName session,
            in boolean forceOverride,
```

```
            in cmiIntermarketCallback::CMIIntermarketOrderStatusConsumer
imOrderStatusListener,
            in cmiIntermarketCallback::CMINBBOAgentSessionAdmin
nbboAgentSessionAdmin )
                raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::DataValidationException,
                    exceptions::TransactionFailedException
                );

        void unregisterAgent(
            in cmiProduct::ClassKey classKey,
            in cmiSession::TradingSessionName session,
            in cmiIntermarketCallback::CMIIntermarketOrderStatusConsumer
imOrderStatusListener,
            in cmiIntermarketCallback::CMINBBOAgentSessionAdmin
nbboAgentSessionAdmin )
                raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::DataValidationException,
                    exceptions::TransactionFailedException
                );
      };
    interface IntermarketUserSessionManager
    {

        IntermarketQuery getIntermarketQuery()
            raises(
                exceptions::SystemException,
                exceptions::CommunicationException,
                exceptions::AuthorizationException
            );

        NBBOAgent getNBBOAgent()
            raises(
                exceptions::SystemException,
                exceptions::CommunicationException,
                exceptions::AuthorizationException
            );
    };

    struct IntermarketSessionManagerStruct
    {
        cmi::UserSessionManager sessionManager;
        IntermarketUserSessionManager imSessionManager;
    };

    interface IntermarketUserAccess
    {

        IntermarketSessionManagerStruct 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
                );

        IntermarketUserSessionManager getIntermarketUserSessionManager(
            in cmi::UserSessionManager session )
                raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::NotFoundException
                );
};
```

cmiIntermarketCallback.idl

```
module cmiIntermarketCallback
{
    interface CMIIntermarketOrderStatusConsumer
    {
        void acceptNewHeldOrder(
            in cmiIntermarketMessages::HeldOrderDetailStruct heldOrder);

        void acceptCancelHeldOrderRequest(
            in cmiProduct::ProductKeysStruct productKeys,
            in cmiIntermarketMessages::HeldOrderCancelRequestStruct
cancelRequestStruct );

        void acceptHeldOrderStatus(
            in cmiIntermarketMessages::HeldOrderDetailStructSequence
heldOrders);

        void acceptHeldOrderCanceledReport(
            in cmiIntermarketMessages::HeldOrderCancelReportStruct
canceledReport );

          void acceptHeldOrderFilledReport(
            in cmiIntermarketMessages::HeldOrderFilledReportStruct
filledReport );

        void acceptFillRejectReport(
            in cmiIntermarketMessages::OrderFillRejectStruct
orderFillReject);
    };
```

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

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

};
```

cmiIntermarketMessages.idl

```
cmiIntermarketMessages.idl
module cmiIntermarketMessages
{

    typedef string ExecutionId;
    typedef short FillRejectReason;
    typedef short HandlingInstruction;
    typedef short ExchangeMarketInfoType;

    struct ExchangeMarketStruct
    {
        cmiIntermarketMessages::ExchangeMarketInfoType marketInfoType;
        cmiUtil::PriceStruct bestBidPrice;
        cmiMarketData::ExchangeVolumeStructSequence bidExchangeVolumes;
        cmiUtil::PriceStruct bestAskPrice;
        cmiMarketData::ExchangeVolumeStructSequence askExchangeVolumes;
    };


    typedef sequence <ExchangeMarketStruct> ExchangeMarketStructSequence;

    struct HeldOrderStruct
    {
       cmiOrder::OrderStruct order;
       cmiIntermarketMessages::ExchangeMarketStructSequence
currentMarketBest;
    };

    typedef sequence <HeldOrderStruct> HeldOrderStructSequence;

    struct HeldOrderDetailStruct
    {
        cmiProduct::ProductNameStruct productInformation;
        cmiUtil::UpdateStatusReason statusChange;
        cmiIntermarketMessages::HeldOrderStruct heldOrder;
    };
```

```
   typedef sequence <HeldOrderDetailStruct> HeldOrderDetailStructSequence;

    struct OrderReminderStruct
    {
        cmiOrder::OrderIdStruct reminderId;
        string reminderReason;
        cmiUtil::DateTimeStruct timeSent;
    };

    struct HeldOrderCancelRequestStruct
    {
        cmiUtil::CboeIdStruct cancelReqId;
        cmiOrder::CancelRequestStruct cancelRequest;
    };
    typedef sequence <HeldOrderCancelRequestStruct>
HeldOrderCancelRequestStructSequence;

    struct HeldOrderCancelReportStruct
    {
        HeldOrderDetailStruct heldOrderDetail;
        cmiUtil::CboeIdStruct cancelReqId;
        cmiOrder::CancelReportStruct cancelReport;
    };

    struct HeldOrderFilledReportStruct
    {
        HeldOrderDetailStruct heldOrderDetail;
        cmiOrder::FilledReportStructSequence filledReport;
    };

    struct CurrentIntermarketBestStruct {
         cmiUser::Exchange exchange;
         cmiSession::ProductState marketCondition;
         cmiUtil::PriceStruct bidPrice;
         long bidVolume;
         cmiUtil::PriceStruct askPrice;
         long askVolume;
         cmiUtil::TimeStruct sentTime;
    };
    typedef sequence <CurrentIntermarketBestStruct>
CurrentIntermarketBestStructSequence;

    struct CurrentIntermarketStruct
    {
         cmiProduct::ProductKeysStruct productKeys;
         CurrentIntermarketBestStructSequence otherMarketsBest;
    };
    typedef sequence <CurrentIntermarketStruct>
CurrentIntermarketStructSequence;

    struct FillRejectStruct
    {
         cmiUtil::CboeIdStruct tradeId;
         cmiOrder::OrderStruct order;
```

```
            long transactionSequenceNumber;
            cmiIntermarketMessages::FillRejectReason rejectReason;
            string extensions;
            string text;
        };
        typedef sequence <FillRejectStruct> FillRejectStructSequence;

        struct OrderFillRejectStruct
        {
            cmiOrder::OrderDetailStruct rejectedFillOrder;
            cmiIntermarketMessages::FillRejectStructSequence fillRejectReports;
        };
        typedef sequence <OrderFillRejectStruct> OrderFillRejectStructSequence;

        struct FillRejectRequestStruct
        {
            cmiOrder::OrderIdStruct orderId;
            cmiIntermarketMessages::FillRejectReason rejectReason;
            string extensions;
            string text;
        };


    };
```

cmiErrorCodes.idl

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

        // 1500 series for additions made for linkage support
        const exceptions::ErrorCode INVALID_EXCHANGE = 1500;
    };

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

    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;
        };
    };
```

Example Program Changes

Java Examples

Common

No change between Version 2.0.1 and Version 2.1.

Example 1

No change between Version 2.0.1 and Version 2.1.

Example 2

No change between Version 2.0.1 and Version 2.1.

Example 3

No change between Version 2.0.1 and Version 2.1.

Example 4

No change between Version 2.0.1 and Version 2.1.

Example 5

No change between Version 2.0.1 and Version 2.1.

Example 6

No change between Version 2.0.1 and Version 2.1.

Example 7

New example for Intermarket Price Protection (IPP).

Example 8

New example for IPP

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.

Simulator Changes

Two new examples for IPP.