CBOE Application Programming Interface

CBOE API Version 9.0.2 - Release Notes

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

CBOE PROPRIETARY INFORMATION

July 15, 2011

Document #[API-00]

Front Matter

Disclaimer

Support and Questions Regarding This Document

Questions regarding this document can be directed to The Chicago Board Options Exchange at 312.786.7300 or via e-mail: api@cboe.com.

The latest version of this document can be found at the CBOE web site: http://systems.cboe.com/webAPI.

Table of Contents

Overview

This document highlights upcoming changes in the new release of the CMi API, Version 9.0.2. Version 9.0.2 supports new CMi interfaces (CMi V10), new IDL and overall documentation changes. The sections below detail the changes in this release. Your feedback or questions regarding this document should be sent to api@cboe.com.

CMi API V9.0.2 Highlights

The upcoming CBOEdirect software release includes new CMi V10 interfaces that support cancel/replace of Light orders and a new subscription methods for quote fills. New contingency types and activity reason messages for Wash Trade Prevention are also included in this release.

CMi V10 Interfaces

Cancel/Replace for Light Orders

The CMi V10 interfaces provide the mechanism for canceling and replacing Light orders.

Session Management

The new CMi V10 interfaces will be made available as extensions to the existing CMi on the existing CAS. A CMi user can gain access to the enhanced interfaces by getting a reference to the UserSessionManagerV10. This reference can be obtained through logon using the UserAccessV10 interface. The new logon method takes exactly the same parameters as the old CMi's logon method and returns a newly enhanced SessionManagerV10. The UserAccessV10 object will be made available as an alternate IOR link on the HTTP port that the CAS is publishing on.

For the logonV2 method, the userHeartbeatTimeout value should be set from 3 (seconds) to 20 (seconds). If the heartbeat timeout is set to less than 3, the system will default the timeout to 3 seconds. If the heartbeat timeout is set to greater than 20, the system will default the timeout to 20 seconds.

```
interface UserAccessV10
{
    UserSessionManagerV10 logon(
    in cmiUser::UserLogonStruct logonStruct,
    in cmiSession::LoginSessionType sessionType,
    in cmiCallback::CMIUserSessionAdmin clientListener,
    in boolean gmdTextMessaging )
                raises(
                exceptions::SystemException,
                exceptions::CommunicationException,
                exceptions::AuthorizationException,
                    exceptions::AuthenticationException,
                    exceptions::DataValidationException,
                    exceptions::NotFoundException
                );
            UserSessionManagerV10 logonV2(
                in cmiUser::UserLogonStruct logonStruct,
                in cmiSession::LoginSessionType sessionType,
                in cmiCallback::CMIUserSessionAdmin clientListener,
                in boolean gmdTextMessaging,
                    in long userHeartbeatTimeout )
                        raises(
                        exceptions::SystemException,
                        exceptions::CommunicationException,
                        exceptions::AuthorizationException,
                        exceptions::AuthenticationException,
                        exceptions::DataValidationException,
                        exceptions::NotFoundException
                        );

        interface UserSessionManagerV10 : cmiV9::UserSessionManagerV9
            {
                cmiV10::OrderEntry   getOrderEntryV10()
            raises(
                exceptions::SystemException,
                exceptions::CommunicationException,
                exceptions::AuthorizationException,
                exceptions::AuthenticationException,
                exceptions::NotFoundException
                );
```

Cancel/Replace Light Orders

A light order cancel/replace is created by calling the acceptLightOrderCancelReplaceRequest method in the cmiV9:OrderEntry interface. The acceptLightOrderCancelReplaceRequest method takes the cmiOrder: LightOrderEntryStruct as an argument and returns the cmiOrder::

LightOrderReplaceResultStruct on successful entry. If there is a quantity mismatch, CMi users will receive Activity Reason message, const cmiUtil::ActivityReason MISMATCHED_QUANTITY = 907. The user should be aware of the following scenarios:

- If a user tries to cancel a quantity that is greater than the remaining quantity (suggesting an in-flight fill), CBOEdirect will cancel the remaining quantity and cancel the replace order.
- If a user tries to cancel a quantity that is less than the remaining on the order and replace it, the cancel request quantity will be canceled and the replace order will also be canceled.
- Light Order Cancel Replace of IOC contingency orders is not supported. The Cancel/Replace request with this contingency type will be rejected.

```
    interface OrderEntry: cmiV9::OrderEntry
    {
            cmiOrder:: LightOrderReplaceResultStruct
    acceptLightOrderCancelReplaceRequest(
          in long originalOrderHighId,
          in long originalOrderLowId,
          in cmiSession::TradingSessionName activeSession,
          in long quantityToCancel,
          in string userAssignedCancelReplaceId,
               in cmiOrder::LightOrderEntryStruct replaceOrder
               )
               raises(
                 exceptions::SystemException,
                 exceptions::CommunicationException,
                 exceptions::AuthorizationException,
                 exceptions::DataValidationException,
                 exceptions::NotAcceptedException,
                 exceptions::TransactionFailedException
               );
 };
    module cmiOrder
            {
            struct LightOrderEntryStruct
                 {
                   string branch;
```

```
                    long branchSequenceNumber;
                    long originalQuantity;
                    double Price;
                    cmiProduct::ProductKey productKey;
                    cmiUtil::Side side;
                    cmiOrder::PositionEffect positionEffect;
                    cmiOrder::Coverage coverage;
                    boolean isNBBOProtected;
                    boolean isIOC;
                    cmiOrder::OriginType orderOriginType;
                    cmiUser::Exchange cmtaExchange;
                    string cmtaFirmNumber;
                    string pdpm;
                    string userAssignedId;
                    cmiSession::TradingSessionName activeSession;
                  };
struct LightOrderReplaceResultStruct
   {
          cmiOrder::LightOrderResultStruct originalOrder;
          cmiOrder::LightOrderResultStruct newOrder;
   };
   typedef sequence <LightOrderReplaceResultStruct>
LightOrderReplaceResultStructSequence;


};
```

Quote Fill Messages

In a later release of CBOEdirect, CMi users, accessing the CMi V10 interfaces, will be able to subscribe to receive only quote fill messages. Buy subscribing to receive only quote fill messages, the queuing of quote fill messages behind other quote status messages is avoided. New methods are available for the subscription of quote fill messages using the cmiV10.Quote which is part of the V10 Session Manager.

Session Management

A CMi user can gain access to the enhanced interfaces by getting a reference to the UserSessionManagerV10. This reference can be obtained through logon using the UserAccessV10 interface. The new logon method takes exactly the same parameters as the old

CMi’s logon method and returns a newly enhanced SessionManagerV10. The UserAccessV10 object will be made available as an alternate IOR link on the HTTP port that the CAS is publishing on.

```
interface UserAccessV10
   {
         UserSessionManagerV10 logon(
         in cmiUser::UserLogonStruct logonStruct,
         in cmiSession::LoginSessionType sessionType,
         in cmiCallback::CMIUserSessionAdmin clientListener,
         in boolean gmdTextMessaging )
                     raises(
                     exceptions::SystemException,
                     exceptions::CommunicationException,
                     exceptions::AuthorizationException,
                     exceptions::AuthenticationException,
                     exceptions::DataValidationException,
                     exceptions::NotFoundException
                     );

         UserSessionManagerV10 logonV2(
         in cmiUser::UserLogonStruct logonStruct,
         in cmiSession::LoginSessionType sessionType,
         in cmiCallback::CMIUserSessionAdmin clientListener,
         in boolean gmdTextMessaging,
               in long userHeartbeatTimeout )
                     raises(
                     exceptions::SystemException,
                     exceptions::CommunicationException,
                     exceptions::AuthorizationException,
                     exceptions::AuthenticationException,
                     exceptions::DataValidationException,
                     exceptions::NotFoundException
                     );
```

```
    };


        interface UserSessionManagerV10 : cmiV9::UserSessionManagerV9
                  {
                          cmiV10::Quote getQuoteV10()
                raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException
                );
            };
```

Subscribing/Unsubscribing for Quote Messages

The following methods support subscribing and unsubscribing for quote messages. Firms should be aware that even if the CMiQuoteStatusConsumer object (which supports all quote status methods) is used, only the appropriate method will be called depending on the type of subscription. For example, if a user subscribes for the subscribeQuoteFillStatus method then the only messages the user should expect to receive would be quote fill messages. The remaining methods will never be called from this subscription.

```
        interface Quote: cmiV7::Quote
                    {
                        void subscribeQuoteFillStatus(
                            in cmiCallbackV2::CMIQuoteStatusConsumer clientListener,
                            in boolean publishOnSubscribe,
                            in boolean includeUserInitiatedStatus,
                            in boolean gmdCallback)
                                raises(
                                    exceptions::SystemException,
                                    exceptions::CommunicationException,
                                    exceptions::DataValidationException,
                                    exceptions::AuthorizationException
                                );

                        void unsubscribeQuoteFillStatus(
                            in cmiCallbackV2::CMIQuoteStatusConsumer clientListener)
                                raises(
                                    exceptions::SystemException,
                                    exceptions::CommunicationException,
                                    exceptions::AuthorizationException,
                                    exceptions::DataValidationException
                                );

                        void subscribeQuoteFillStatusForClass (
                            in cmiProduct::ClassKey classKey,
                            in boolean publishOnSubscribe,
                            in boolean includeUserInitiatedStatus,
                            in cmiCallbackV2::CMIQuoteStatusConsumer clientListener,
                            in boolean gmdCallback)
                                raises(
                                    exceptions::SystemException,
                                    exceptions::CommunicationException,
                                    exceptions::DataValidationException,
                                    exceptions::AuthorizationException
                                );

                        void unsubscribeQuoteFillStatusForClass(
                            in cmiProduct::ClassKey classKey,
                            in cmiCallbackV2::CMIQuoteStatusConsumer clientListener)
                                raises(
                                    exceptions::SystemException,
                                    exceptions::CommunicationException,
                                    exceptions::AuthorizationException,
                                    exceptions::DataValidationException
                                );

                         void subscribeQuoteStatusWithoutFill(
                            in cmiCallbackV2::CMIQuoteStatusConsumer clientListener,
                            in boolean publishOnSubscribe,
                            in boolean includeUserInitiatedStatus,
                            in boolean gmdCallback)
                                raises(
                                    exceptions::SystemException,
                                    exceptions::CommunicationException,
                                    exceptions::DataValidationException,
                                    exceptions::AuthorizationException
                                );

                        void unsubscribeQuoteStatusWithoutFill(
                            in cmiCallbackV2::CMIQuoteStatusConsumer clientListener)
                                raises(
                                    exceptions::SystemException,
                                    exceptions::CommunicationException,
                                    exceptions::AuthorizationException,
                                    exceptions::DataValidationException
                                );

                        void subscribeQuoteStatusWithoutFillForClass (
                            in cmiProduct::ClassKey classKey,
                            in boolean publishOnSubscribe,
                            in boolean includeUserInitiatedStatus,
                            in cmiCallbackV2::CMIQuoteStatusConsumer clientListener,
                            in boolean gmdCallback)
                                raises(
                                    exceptions::SystemException,
                                    exceptions::CommunicationException,
                                    exceptions::DataValidationException,
                                    exceptions::AuthorizationException
                                );

                        void unsubscribeQuoteStatusWithoutFillForClass(
                            in cmiProduct::ClassKey classKey,
                            in cmiCallbackV2::CMIQuoteStatusConsumer clientListener)
                                raises(
                                    exceptions::SystemException,
                                    exceptions::CommunicationException,
                                    exceptions::AuthorizationException,
                                    exceptions::DataValidationException
                                );
                              };
```

Wash Trade Prevention

New Order Contingency Type

The next CBOEdirect release will include a new contingency type for Wash Trade Prevention, **const cmiOrder::ContingencyType WTP = 35; // Wash Trade Prevention.** When marked with this contingency, the order will be cancelled if it is about to trade against another order or quote from the same user. Additionally, the resting quote or order will be cancelled.

New Activity Reason Message

CMi users will receive a new Activity Reason message for Wash Trade Prevention, **const cmiUtil::ActivityReason WASH_TRADE_PREVENTION = 906.** This message will notify the user that their order or quote was cancelled from the system because it was about to trade against another order or quote from the same user.

IDL Interfaces

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

```
module cmiV10
{
        //---------------------------------------------------------------------------------------------------------
        //          Light Order Cancel Replacement Interface
        //---------------------------------------------------------------------------------------------------------

  interface OrderEntry: cmiV9::OrderEntry
  {

                        cmiOrder:: LightOrderReplaceResultStruct
acceptLightOrderCancelReplaceRequest(
                in long originalOrderHighId,
                in long originalOrderLowId,
                in cmiSession::TradingSessionName activeSession,
            in long quantityToCancel,
            in string userAssignedCancelReplaceId,
                    in cmiOrder::LightOrderEntryStruct replaceOrder
                    )
                    raises(
                      exceptions::SystemException,
                      exceptions::CommunicationException,
                      exceptions::AuthorizationException,
                      exceptions::DataValidationException,
                      exceptions::NotAcceptedException,
                      exceptions::TransactionFailedException
                    );

  };

interface Quote: cmiV7::Quote
         {
           void subscribeQuoteFillStatus(
             in cmiCallbackV2::CMIQuoteStatusConsumer clientListener,
             in boolean publishOnSubscribe,
             in boolean includeUserInitiatedStatus,
             in boolean gmdCallback)
               raises(
                 exceptions::SystemException,
                 exceptions::CommunicationException,
                 exceptions::DataValidationException,
                 exceptions::AuthorizationException
               );

           void unsubscribeQuoteFillStatus(
             in cmiCallbackV2::CMIQuoteStatusConsumer clientListener)
               raises(
                 exceptions::SystemException,
                 exceptions::CommunicationException,
                 exceptions::AuthorizationException,
                 exceptions::DataValidationException
```

```
        );


    void subscribeQuoteFillStatusForClass (
      in cmiProduct::ClassKey classKey,
      in boolean publishOnSubscribe,
      in boolean includeUserInitiatedStatus,
      in cmiCallbackV2::CMIQuoteStatusConsumer clientListener,
      in boolean gmdCallback)
        raises(
          exceptions::SystemException,
          exceptions::CommunicationException,
          exceptions::DataValidationException,
          exceptions::AuthorizationException
        );

    void unsubscribeQuoteFillStatusForClass(
      in cmiProduct::ClassKey classKey,
      in cmiCallbackV2::CMIQuoteStatusConsumer clientListener)
        raises(
          exceptions::SystemException,
          exceptions::CommunicationException,
          exceptions::AuthorizationException,
          exceptions::DataValidationException
        );


    void subscribeQuoteStatusWithoutFill(
      in cmiCallbackV2::CMIQuoteStatusConsumer clientListener,
      in boolean publishOnSubscribe,
      in boolean includeUserInitiatedStatus,
      in boolean gmdCallback)
        raises(
          exceptions::SystemException,
          exceptions::CommunicationException,
          exceptions::DataValidationException,
            exceptions::AuthorizationException
        );

void unsubscribeQuoteStatusWithoutFill(
    in cmiCallbackV2::CMIQuoteStatusConsumer clientListener)
        raises(
            exceptions::SystemException,
            exceptions::CommunicationException,
            exceptions::AuthorizationException,
            exceptions::DataValidationException
        );



void subscribeQuoteStatusWithoutFillForClass (
    in cmiProduct::ClassKey classKey,
    in boolean publishOnSubscribe,
    in boolean includeUserInitiatedStatus,
    in cmiCallbackV2::CMIQuoteStatusConsumer clientListener,
    in boolean gmdCallback)
        raises(
            exceptions::SystemException,
            exceptions::CommunicationException,
            exceptions::DataValidationException,
            exceptions::AuthorizationException
        );

void unsubscribeQuoteStatusWithoutFillForClass(
    in cmiProduct::ClassKey classKey,
    in cmiCallbackV2::CMIQuoteStatusConsumer clientListener)
        raises(
            exceptions::SystemException,
            exceptions::CommunicationException,
            exceptions::AuthorizationException,
            exceptions::DataValidationException
        );
      };
```

```
interface UserSessionManagerV10 : cmiV9::UserSessionManagerV9
      {
              cmiV10::OrderEntry   getOrderEntryV10()
      raises(
              exceptions::SystemException,
              exceptions::CommunicationException,
              exceptions::AuthorizationException,
              exceptions::AuthenticationException,
              exceptions::NotFoundException
              );

              cmiV10::Quote getQuoteV10()
                    raises(
                      exceptions::SystemException,
                      exceptions::CommunicationException,
                      exceptions::AuthorizationException

};

interface UserAccessV10
{
      UserSessionManagerV10 logon(
      in cmiUser::UserLogonStruct logonStruct,
      in cmiSession::LoginSessionType sessionType,
      in cmiCallback::CMIUserSessionAdmin clientListener,
      in boolean gmdTextMessaging )
                    raises(
                    exceptions::SystemException,
                    exceptions::CommunicationException,
                    exceptions::AuthorizationException,
                    exceptions::AuthenticationException,
                    exceptions::DataValidationException,
                    exceptions::NotFoundException
                    );
```

```
        UserSessionManagerV10 logonV2(
                in cmiUser::UserLogonStruct logonStruct,
                in cmiSession::LoginSessionType sessionType,
                in cmiCallback::CMIUserSessionAdmin clientListener,
                in boolean gmdTextMessaging,
                        in long userHeartbeatTimeout )
                                raises(
                                exceptions::SystemException,
                                exceptions::CommunicationException,
                                exceptions::AuthorizationException,
                                exceptions::AuthenticationException,
                                exceptions::DataValidationException,
                                exceptions::NotFoundException
                                );
 };

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;
          typedef short AuctionType;
          typedef short AuctionState;
          typedef short OrderMaintenanceType;
          typedef char OrderType;

          typedef sequence <cmiOrder::OriginType> OriginTypeSequence;
          typedef sequence <cmiOrder::AuctionType> AuctionTypeSequence;
          typedef sequence <cmiOrder::OrderType> OrderTypeSequence;
```

```
#pragma use_factory_for_struct ON
    struct OrderContingencyStruct
    {
        cmiOrder::ContingencyType type;
        cmiUtil::PriceStruct price;
        long volume;
    };
#pragma use_factory_for_struct OFF

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

#pragma use_factory_for_struct OFF

    typedef sequence <OrderIdStruct> OrderIdStructSequence;

    struct ORDOrderStruct
    {
        OrderIdStruct orderId;
        cmiOrder::OrderState state;
        long bookedQuantity;
    };
    typedef sequence <ORDOrderStruct> ORDOrderStructSequence;

    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 LegOrderEntryStructV2
    {
        cmiOrder::LegOrderEntryStruct legOrderEntry;
        cmiUtil::Side side;
              string extensions;
    };
    typedef sequence <LegOrderEntryStructV2> LegOrderEntryStructV2Sequence;



    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;



#pragma use_factory_for_struct ON
    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;
    };
#pragma use_factory_for_struct OFF
    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;
        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;

    typedef short MatchType; // can be auto-match, fixed limit price match, or guaranteed auction
starting price match

    struct AuctionStruct
    {
            cmiSession::TradingSessionName sessionName;
            cmiProduct::ClassKey classKey;
            cmiProduct::ProductType productType;
            cmiProduct::ProductKey productKey;
            cmiUtil::CboeIdStruct auctionId;
            cmiOrder::AuctionType auctionType;
            cmiOrder::AuctionState auctionState;
            cmiUtil::Side side;
            long auctionQuantity;
            cmiUtil::PriceStruct startingPrice;
            cmiOrder::ContingencyType auctionedOrderContingencyType;
            cmiUtil::TimeStruct entryTime;
            string extensions;
    };
    typedef sequence <AuctionStruct> AuctionStructSequence;

    // For internalization Orders call return
    struct OrderResultStruct
    {
        cmiOrder::OrderIdStruct orderId;
        cmiUtil::OperationResultStruct result;
    };
    typedef sequence <OrderResultStruct> OrderResultStructSequence;


    struct InternalizationOrderResultStruct
    {
        cmiOrder::OrderResultStruct primaryOrderResult;
        cmiOrder::OrderResultStruct matchOrderResult;
    };
    typedef sequence <InternalizationOrderResultStruct>
InternalizationOrderResultStructSequence;


    struct AuctionSubscriptionResultStruct
    {
        cmiOrder::AuctionType auctionType;
        cmiUtil::OperationResultStruct subscriptionResult;
    };
    typedef sequence <AuctionSubscriptionResultStruct>
AuctionSubscriptionResultStructSequence;


    struct OrderResultStructV2
    {
        cmiOrder::OrderStruct order;
        cmiUtil::OperationResultStruct result;
    };
    typedef sequence <OrderResultStructV2> OrderResultStructV2Sequence;


    struct InternalizationOrderResultStructV2
    {
        cmiOrder::OrderResultStructV2 primaryOrderResult;
        cmiOrder::OrderResultStructV2 matchOrderResult;
    };
    typedef sequence <InternalizationOrderResultStructV2>
InternalizationOrderResultStructV2Sequence;


    struct CrossOrderStruct
    {
             cmiOrder::OrderStruct buySideOrder;
        cmiOrder::OrderStruct sellSideOrder;
};
struct LightOrderEntryStruct
{
    string branch;
    long branchSequenceNumber;
    long originalQuantity;
    double Price;
    cmiProduct::ProductKey productKey;
    cmiUtil::Side side;
    cmiOrder::PositionEffect positionEffect;
    cmiOrder::Coverage coverage;
    boolean isNBBOProtected;
    boolean isIOC;
    cmiOrder::OriginType orderOriginType;
    cmiUser::Exchange cmtaExchange;
    string cmtaFirmNumber;
    string pdpm;
    string userAssignedId;
    cmiSession::TradingSessionName activeSession;
};

struct LightOrderResultStruct
{
    string branch;
    long branchSequenceNumber;
    long orderHighId;
    long orderLowId;
    cmiUtil::Side side;
    long leavesQuantity;
    long tradedQuantity;
    long cancelledQuantity;
    cmiUtil::ActivityReason reason;
    cmiUtil::DateTimeStruct time;
};
  typedef sequence <LightOrderResultStruct> LightOrderResultStructSequence;
```

```
            typedef sequence <LightOrderResultStruct> LightOrderResultStructSequence;

               struct LightOrderReplaceResultStruct
               {
                       cmiOrder::LightOrderResultStruct originalOrder;
                       cmiOrder::LightOrderResultStruct newOrder;
               };
               typedef sequence <LightOrderReplaceResultStruct>
            LightOrderReplaceResultStructSequence;

            };
```

module cmiConstants

```
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
      const cmiOrder::ContingencyType AUCTION_RESPONSE = 14; // Auction response order
      const cmiOrder::ContingencyType INTERMARKET_SWEEP = 15; // Intermarket sweep (ISO)
      const cmiOrder::ContingencyType RESERVE = 16; // Reserve order
      const cmiOrder::ContingencyType MIDPOINT_CROSS    = 17; // Mid Point Cross
      const cmiOrder::ContingencyType CROSS             = 18; // Cross
      const cmiOrder::ContingencyType TIED_CROSS        = 19; // Tied cross
      const cmiOrder::ContingencyType AUTOLINK_CROSS    = 20; // Auto link cross
```

```
        const cmiOrder::ContingencyType AUTOLINK_CROSS_MATCH = 21; // Auto link cross
        const cmiOrder::ContingencyType CROSS_WITHIN      = 22;
        const cmiOrder::ContingencyType TIED_CROSS_WITHIN = 23;
        const cmiOrder::ContingencyType STOCK_ODD_LOT_NBBO_ONLY = 24;
        const cmiOrder::ContingencyType NBBO_FLASH_THEN_CANCEL = 25;
        const cmiOrder::ContingencyType DO_NOT_ROUTE = 26;
        const cmiOrder::ContingencyType NBBO_FLASH_RESPONSE = 27;
        const cmiOrder::ContingencyType INTERMARKET_SWEEP_BOOK = 28; // Intermarket Sweep
Book (ISB)
        const cmiOrder::ContingencyType BID_PEG_CROSS = 29;
        const cmiOrder::ContingencyType OFFER_PEG_CROSS = 30;
        const cmiOrder::ContingencyType TIED_CROSS_SWEEP = 31;
        const cmiOrder::ContingencyType CASH_CROSS = 32; // cash settlement cross
        const cmiOrder::ContingencyType NEXT_DAY_CROSS = 33; // Next day settlement cross
        const cmiOrder::ContingencyType TWO_DAY_CROSS  = 34; // Two day settlement cross
```

const cmiOrder::ContingencyType WTP = 35; // Wash Trade Prevention

```
interface ActivityReasons
   {
        const cmiUtil::ActivityReason NOTHING_DONE = 1;
        const cmiUtil::ActivityReason USER = 2;
        const cmiUtil::ActivityReason SYSTEM = 3;
        const cmiUtil::ActivityReason LOST_CONNECTION = 4;
        const cmiUtil::ActivityReason INSUFFICIENT_QUANTITY = 5;
        const cmiUtil::ActivityReason SPECIAL_ADJUSTMENT = 6;
        const cmiUtil::ActivityReason QRM_REMOVED = 7;
        const cmiUtil::ActivityReason INSUFFICIENT_QUANTITY_BUY_SIDE  = 8;
        const cmiUtil::ActivityReason INSUFFICIENT_QUANTITY_SELL_SIDE  = 9;
        const cmiUtil::ActivityReason QUOTE_UPDATE_CONTROL =10;
        // acceptServerFailure event would have following reason
        const cmiUtil::ActivityReason FAILOVER= 11;
        const cmiUtil::ActivityReason QUOTE_IN_TRIGGER =12;
        const cmiUtil::ActivityReason INVALID_SESSION_ID =13;
        const cmiUtil::ActivityReason SAL_IN_PROGRESS = 14;
        const cmiUtil::ActivityReason CROSS_IN_PROGRESS = 15;
        const cmiUtil::ActivityReason INVALID_NBBO = 16;
        const cmiUtil::ActivityReason NOT_WITHIN_NBBO = 17;
        const cmiUtil::ActivityReason TRADE_THROUGH_CBOE = 18;
        const cmiUtil::ActivityReason INSUFFICIENT_CUSTOMER_ORDER_QUANTITY = 19;
        const cmiUtil::ActivityReason INSUFFICIENT_CROSS_ORDER_SIZE = 20;
        const cmiUtil::ActivityReason INSUFFICIENT_CROSS_ORDER_DOLLAR_AMOUNT = 21;
        const cmiUtil::ActivityReason SELL_SHORT_RULE_VIOLATION = 22;
        // acceptUserActivityTimeout (UIM) event would have the following reason:
        const cmiUtil::ActivityReason NO_USER_ACTIVITY = 23;
        const cmiUtil::ActivityReason CANCEL_ON_RSS = 24;

        // acceptServerFailure will have the following reason code in case of CDX fast failover
        const cmiUtil::ActivityReason QUOTE_UPDATES_REQUESTED = 25;

        // The following are used for Linkage

        const cmiUtil::ActivityReason BROKER_OPTION = 100;
        const cmiUtil::ActivityReason CANCEL_PENDING = 101;
        const cmiUtil::ActivityReason CROWD_TRADE = 102;
        const cmiUtil::ActivityReason DUPLICATE_ORDER = 103;
        const cmiUtil::ActivityReason EXCHANGE_CLOSED = 104;
        const cmiUtil::ActivityReason GATE_VIOLATION = 105;
        const cmiUtil::ActivityReason INVALID_ACCOUNT = 106;
        const cmiUtil::ActivityReason INVALID_AUTOEX_VALUE = 107;
        const cmiUtil::ActivityReason INVALID_CMTA = 108;
        const cmiUtil::ActivityReason INVALID_FIRM = 109;
        const cmiUtil::ActivityReason INVALID_ORIGIN_TYPE = 110;
        const cmiUtil::ActivityReason INVALID_POSITION_EFFECT = 111;
        const cmiUtil::ActivityReason INVALID_PRICE = 112;
        const cmiUtil::ActivityReason INVALID_PRODUCT = 113;
        const cmiUtil::ActivityReason INVALID_PRODUCT_TYPE = 114;
        const cmiUtil::ActivityReason INVALID_QUANTITY = 115;
        const cmiUtil::ActivityReason INVALID_SIDE = 116;
        const cmiUtil::ActivityReason INVALID_SUBACCOUNT = 117;
        const cmiUtil::ActivityReason INVALID_TIME_IN_FORCE = 118;
        const cmiUtil::ActivityReason INVALID_USER = 119;
```

```
 const cmiUtil::ActivityReason LATE_PRINT = 120;
 const cmiUtil::ActivityReason NOT_FIRM = 121;
 const cmiUtil::ActivityReason MISSING_EXEC_INFO = 122;
 const cmiUtil::ActivityReason NO_MATCHING_ORDER = 123;
 const cmiUtil::ActivityReason NON_BLOCK_TRADE = 124;
 const cmiUtil::ActivityReason NOT_NBBO = 125;
 const cmiUtil::ActivityReason COMM_DELAYS = 126;
 const cmiUtil::ActivityReason ORIGINAL_ORDER_REJECTED = 127;
 const cmiUtil::ActivityReason OTHER = 128;
 const cmiUtil::ActivityReason PROCESSING_PROBLEMS = 129;
 const cmiUtil::ActivityReason PRODUCT_HALTED = 130;
 const cmiUtil::ActivityReason PRODUCT_IN_ROTATION = 131;
 const cmiUtil::ActivityReason STALE_EXECUTION = 132;
 const cmiUtil::ActivityReason STALE_ORDER = 133;
 const cmiUtil::ActivityReason ORDER_TOO_LATE = 134;
 const cmiUtil::ActivityReason TRADE_BUSTED = 135;
 const cmiUtil::ActivityReason TRADE_REJECTED = 136;
 const cmiUtil::ActivityReason ORDER_TIMEOUT = 141;
 const cmiUtil::ActivityReason REJECTED_LINKAGE_TRADE  = 170;
 const cmiUtil::ActivityReason SATISFACTION_ORD_REJ_OTHER  = 171;
 const cmiUtil::ActivityReason PRODUCT_SUSPENDED = 172;

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

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

// Force Cancel Orders due to fallback
const cmiUtil::ActivityReason CANCEL_ON_FALLBACK = 800;

 // 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;

//Cancel Reason For Wash Trade Prevention.
        const cmiUtil::ActivityReason WASH_TRADE_PREVENTION = 906;
                  };
//Light Order Cancel Replace Reject code.
              const cmiUtil::ActivityReason MISMATCHED_QUANTITY = 907;


    };
```

Document Changes

API-01

- No changes

API-02

- Incorporated a new section for CMi V10

API-03

- Added the new CMi V10 interfaces
- Included the new order contingency type for Wash Trade Prevention

  const cmiOrder::ContingencyType WTP = 35; // Wash Trade Prevention
- Added the following cancel reason for Wash Trade Prevention

  const cmiUtil::ActivityReason WASH_TRADE_PREVENTION = 906;

API-04

- Added struct definitions for the CMi V10 interfaces
- Included the new order contingency type for Wash Trade Prevention

  const cmiOrder::ContingencyType WTP = 35; // Wash Trade Prevention
- Added the following cancel reason for Wash Trade Prevention

  const cmiUtil::ActivityReason WASH_TRADE_PREVENTION = 906;

API-05

- No changes

API-06

- No changes.

API-07

- No changes

API-08

- No changes

CAS-01

- No changes

CAS-02

- No changes

Simulator

- No changes

Test Plan Changes

- No changes