# IDL Changes for Hybrid Projects

## Disseminate Best Quotes to TPF

To reduce the loads on event delivery system and improve the performance, it has been decided that the new way to publish all market information events will be done in a block by classes, which includes following event channels:

1. CurrentMarket
2. Ticker
3. Recap
4. BookDepth

Though only CurrentMarket channel is related to disseminate best quote to TPF, the changes for all the consumers will be outlined below

1. Add following constants to /vobs/dte/client/idl/cmiConstants.idl

```
interface CurrentMarketViewTypes {
        const cmiMarketData::CurrentMarketViewType CONTINGENT =1;
        const cmiMarketData::CurrentMarketViewType NON_CONTINGENT =2;
        const cmiMarketData::CurrentMarketViewType NON_ICM =3;
}
```

2. Add several constants to interface VolumeTypes:

```
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
        const cmiMarketData::VolumeType NO_CONTINGENCY = 5;

        const cmiMarketData::VolumeType NON_ICM_PRIORITY = 6; // Customer
        const cmiMarketData::VolumeType NON_ICM_NON_PRIORITY = 7; // BD
        const cmiMarketData::VolumeType ICM = 8; // In-Crowd Market Maker
};
```

3. Add following structs to /vobs/dte/client/idl/cmiMarketData.idl

```
typedef short CurrentMarketViewType;

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

struct CurrentMarketStructV2{
        cmiProduct::ProductKeysStruct productKeys;
        cmiSession::TradingSessionName sessionName;
        string exchange;
        cmiMarketData::CurrentMarketViewSequence currentMarketViews;
```

```
        cmiUtil::TimeStruct sentTime;
        boolean bidIsMarketBest;
        boolean askIsMarketBest;
        boolean legalMarket;
    }
    typedef sequence <CurrentMarketStructV2> CurrentMarketStructV2Sequence;
```

4. Change CurrentMarketConsumer as followings:

```
    interface CurrentMarketConsumer {

        // the following  interface methods will be deplicated after all CASes which can support
        // block current market publishing will have been rolled out.

        void acceptCurrentMarket(in cmiUtil::LongSequence groups,
                in cmiMarketData::CurrentMarketStruct contingentMarket,
                in cmiMarketData::CurrentMarketStruct nonContingentMarket );

        void acceptNBBO(in cmiUtil::LongSequence groups,
                in cmiMarketData::NBBOStruct nbboStruct );

        void acceptCurrentMarketAndNBBO(in cmiUtil::LongSequence groups,
                in cmiMarketData::CurrentMarketStruct contingentMarket,
                in cmiMarketData::CurrentMarketStruct nonContingentMarket,
                in cmiMarketData::NBBOStruct nbboStruct );

        void acceptExpectedOpeningPrice(in cmiUtil::LongSequence groups,
                in cmiMarketData::ExpectedOpeningPriceStruct expectedOpeningPrice );


        // the new acceptCurrentMarketsForClass is essentially a combination of
        // acceptCurrentMarket, acceptNBBO, and acceptCurrentMarketAndNBBO
        // and publishes in groups by classes. The intended usage of this method is
        // to leave any sequence empty if it does not change when publishing. E.g.
        // if NBBO and NonICMMarkets do not changes, then leave nbbos and nonICMMarkets
        // as empty sequence. If only NBBO changes, then leave all other three sequences as
        // empty sequence.

        void acceptCurrentMarketsForClass(
                in cmiUtil::LongSequence groups,
                in cmiProduct::ClassKey classKey,
                in cmiMarketData::CurrentMarketStructSequence contingentMarkets,
                in cmiMarketData::CurrentMarketStructSequence nonContingentMarkets,
                in cmiMarketData::NBBOStructSequence nbbos,
                in cmiMarketData::CurrentMarketStructV2Sequence markets );

        void acceptExpectedOpeningPricesForClass(
                in cmiUtil::LongSequence groups,
                in cmiProduct::ClassKey classKey,
                in cmiMarketData::ExpectedOpeningPriceStructSequence
                        expectedOpeningPrices );


    };
```

5. Change TickerConsumer as follows:

```
        interface TickerConsumer {

                // the following  interface method will be deplicated after all CASes which can
                //support block  publishing will have been rolled out.
                void acceptTicker(
                        in cmiUtil::LongSequence groups,
                        in marketData::InternalTickerStruct ticker);

                // new method for block publishing by class
                void acceptTickerForClass(
                        in cmiUtil::LongSequence groups,
                        in cmiProduct::ClassKey classKey,
                        in marketData::InternalTickerStructSequence  tickers);

        };
```

6. Change RacapConsumer as follows:

```
        interface RecapConsumer {

                // the following  interface method will be deplicated after all CASes which can
                //support block  publishing will have been rolled out.
                void acceptRecap(
                        in cmiUtil::LongSequence groups,
                        in cmiMarketData::RecapStruct recap);

                // new method for block publishing by class
                void acceptRecapForClass(
                        in cmiUtil::LongSequence groups,
                        in cmiProduct::ClassKey classKey,
                        in cmiMarketData::RecapStructSequence recaps);
        };
```

7. Change BookDepthConsumer as follows:

```
        interface BookDepthConsumer {

                // the following  interface method will be deplicated after all CASes which can
                //support block  publishing will have been rolled out.
                void acceptBookDepth(
                    in cmiUtil::LongSequence groups,
                    in cmiMarketData::BookDepthStruct bookDepth);

                // new method for block publishing by class
                void acceptBookDepthForClass(
                    in cmiUtil::LongSequence groups,
                    in cmiProduct::ClassKey classKey,
                    in cmiMarketData::BookDepthStructSequence bookDepths);
        };
};
```

## Display Last Sale

No IDL changes will be needed for supporting hybrid system

## Define Product

No IDL changes will be needed for supporting hybrid system

## Define Trading Session

No IDL changes will be needed for supporting hybrid system

## Enter SBT Order

No IDL changes will be needed for supporting hybrid system

## Accept Hybrid Order From TPF

## Accept Hybrid Cancel Request from TPF

## Accept Hybrid Cancel/Replace from TPF

1. Add followings constants to /vobs/dte/server/idl/constants.idl

```
interface TradingRestrictions
{
        constant order::TradingRestriction NONE = 1;
        constant order::TradingRestriction NON_ICM_ONLY = 2;
};

interface RemainderHandlingModes
{
        constant order::RemainderHandlingMode BOOK_REMAINDER = 1;
        constant order::RemainderHandlingMode RETURN_REMAINDER = 2;
};

interface MaximumExecutionReasons {
        const order::MaximumExecutionVolumeReason NONE = 1;
};
```

2. Add a wrapper struct to /vobs/dte/server/idl/marketData.idl

```
struct MarketPriceStruct {
        cmiUtil::PriceStruct bestOfTheRestBid;
        cmiUtil::PriceStruct bestOfTheRestAsk;
        cmiUtil::PriceStruct bestBid;
        cmiUtil::PriceStruct bestAsk;
};
```

3. Add OrderHandlingInstructionStruct to /vobs/dte/server/idl/order.idl

```
typedef short TradingRestriction;
typedef short RemainderHandlingMode;
typedef short OrderRoutingReason;
typedef short MaximumExecutionVolumeReason;

struct OrderHandlingInstructionStruct
{
        long maximumExecutionVolume;
        order::MaximumExecutionVolumeReason;
        order::TradingRestriction tradingRestriction;
        order::RemainderHandlingMode remainderHandlingMode;
};
```

4. Add OrderMaintenanceServiceV2 to /vobs/dte/server/idl/OrderMaintenanceService.idl

```
interface OrderMaintenanceServiceV2: OrderMaintenanceServce {

    // accept cancel request from tpf for hybrid order
    void acceptCancel( in string userId, in cmiOrder::CancelRequestStruct cancelRequest, in
    cmiProduct::ProductKeysStruct productKeys )
    raises(
            exceptions::SystemException,
            exceptions::CommunicationException,
            exceptions::DataValidationException,
            exceptions::TransactionFailedException,
             exceptions::NotAcceptedException,
            exceptions::AuthorizationException
     );

    // accept cancel/replace request from tpf for hybrid roder
    cmiOrder::OrderIdStruct acceptCancelReplace( in cmiOrder::CancelRequestStruct
    cancelRequest, in cmiOrder::OrderStruct anOrder )
     raises(
            exceptions::SystemException,
            exceptions::CommunicationException,
            exceptions::DataValidationException,
            exceptions::TransactionFailedException,
            exceptions::NotAcceptedException,
            exceptions::AuthorizationException
            );

    // accept order with handling instruction from tpf
    cmiOrder::OrderIdStruct acceptOrderWithInstructions(
            in cmiOrder::OrderStruct anOrder,
            in order::OrderHandlingInstructionStruct handlingInstruction,
            in util::MarketPriceStruct marketBestPrices )
    raises(
             exceptions::SystemException,
            exceptions::CommunicationException,
            exceptions::DataValidationException,
            exceptions::TransactionFailedException,
            exceptions::NotAcceptedException,
            exceptions::AuthorizationException
```

```
        );
}
```

## Return Hybrid Order to TPF

## Return Hybrid Cancel to TPF

## Return Hybrid Cancel/Replace to TPF

1. Add OrderRoutingReasons constants to /vobs/dte/server/idl/constants.idl

```
interface OrderRoutingReasons
{
        const order::OrderRoutingReason NO_REASON = O;
        const order::OrderRoutingReason SYSTEM_ERROR = 1;
        const order::OrderRoutingReason UNKNOWN_PRODUCT = 2;
        const order::OrderRoutingReason REMAINING_QUANTITY = 3;
        const order::OrderRoutingReason NOT_TRADABLE = 4;
        const order::OrderRoutingReason NOT_BOOKABLE = 5;
        const order::OrderRoutingReason NOT_NBBO = 6;

        // Sending the 'replace' order for a cancel/re to TPF.
        // Does this use the original orsid?  Or does this need to be a different method?
        // The requirements seem to imply that the replaced order should be routed back,
        // not just a cancel report.
        const order::OrderRoutingReason REPLACED = 7;
        const order::OrderRoutingReason REJECTED = 8;
};
```

2. Add OrderRoutingServiceV2 to /vobs/dte/server/idl/OrderRoutingService.idl:

```
interface OrderRoutingServiceV2: OrderRoutingService {

        // reroute hybrid order to tpf
        cmiOrder::OrderIdStruct acceptOrderWithReason(
                in cmiOrder::OrderStruct anOrder,
                in order::OrderRoutingReason reason )
        raises(
                 exceptions::SystemException,
                exceptions::CommunicationException,
                exceptions::DataValidationException,
                exceptions::TransactionFailedException,
                exceptions::NotAcceptedException,
                exceptions::AuthorizationException
        );

        // reroute cancel request for hybrid order to tpf
        void acceptCancelWithReason(
                in string userId,
                in cmiOrder::CancelRequestStruct cancelRequest,
                in cmiProduct::ProductKeysStruct productKeys,
                in order::OrderRoutingReason reason )
        raises(
```

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

        // reroute cancel/replace request for hybrid order to tpf
        cmiOrder::OrderIdStruct acceptCancelReplaceWithReason(
                in cmiOrder::CancelRequestStruct cancelRequest,
                in cmiOrder::OrderStruct anOrder,
                in order::OrderRoutingReason reason )
        raises(
                exceptions::SystemException,
                exceptions::CommunicationException,
                exceptions::DataValidationException,
                exceptions::TransactionFailedException,
                exceptions::NotAcceptedException,
                exceptions::AuthorizationException
        );

}
```

## Display Fill Report

No IDL changes will be needed for supporting hybrid system

## Cancel Order

No IDL changes will be needed for supporting hybrid system

## Disseminate Reports to TPF

Fill and cancel reports do not need an additional “report status” field since the TPF adapter should be able to determine if the report has already been sent to the originator by examining the source and origin codes.

## Disseminate Reports to CTM

No changes needed, the ‘side’ to disseminate can be determined by the TPF adapter by examining the source and origin codes.

## Disseminate Alert to TPF

CBOEDir is required to monitor the last sales of other exchanges and generate tradethrough alerts and send the alert to TPF if other exchanges trade through our published market.

How this is going to be done needs further discussion.

We expect to use the new Alert service to accomplish this. Although code changes will have to be made to the alert service or the SAGUI to ignore satisfaction alerts when copying to the TFL, IDL changes are expected to be be restricted to a new alert message type in constants.idl:

```
interface AlertTypes
 {
     // …
    const alert::AlertType CBOE_SATISFACTION_ALERT = 5;
 }
```