About Trading Components Framework

This documentation describes the procedure to include StreamBase Trading Components in applications you create that can monitor foreign exchange (FX) markets and execute trades at FX venues. It also describes the commands, schemas, properties and constant values used within the framework's modules.

NOTE	The Trading Components Framework was initially released as an optional component in earlier releases of StreamBase and is now integrated into StreamBase Studio. For information on recent changes and updates, see and .

At the highest level, the Trading Components Framework packages its modules into two types of venue-specific handlers:

Market Data Handlers

Modules that access streaming market currency exchange data.

Execution Handlers

Modules that communicate trades with execution venues.

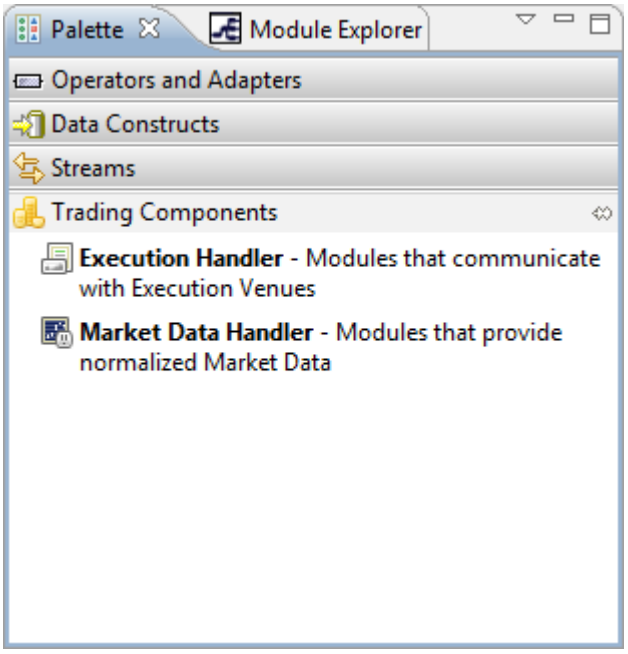
TIBCO has made building FX applications around the handlers as standardized as possible, by providing common port configurations, schemas, and properties. As protocols vary among FX venues, separate handlers are provided for each one. Generally, all you need to do is to provide necessary credentials to connect to venues, and then configure currencies, sides, tenors and other options that you normally need to specify.

The following section describes how to access and start to configure handlers within a StreamBase application. Market data handlers are available for about a dozen venues, and execution handlers are included for most of them. See for a complete list. All adapter properties are parameterized, enabling you to set component properties (generally those of FIX adapters) by setting module parameters, with little need to edit configuration files beyond providing required credentials. Also, a set of common that all handlers use standardize data fields within applications and enable them to transparently communicate with different venues.

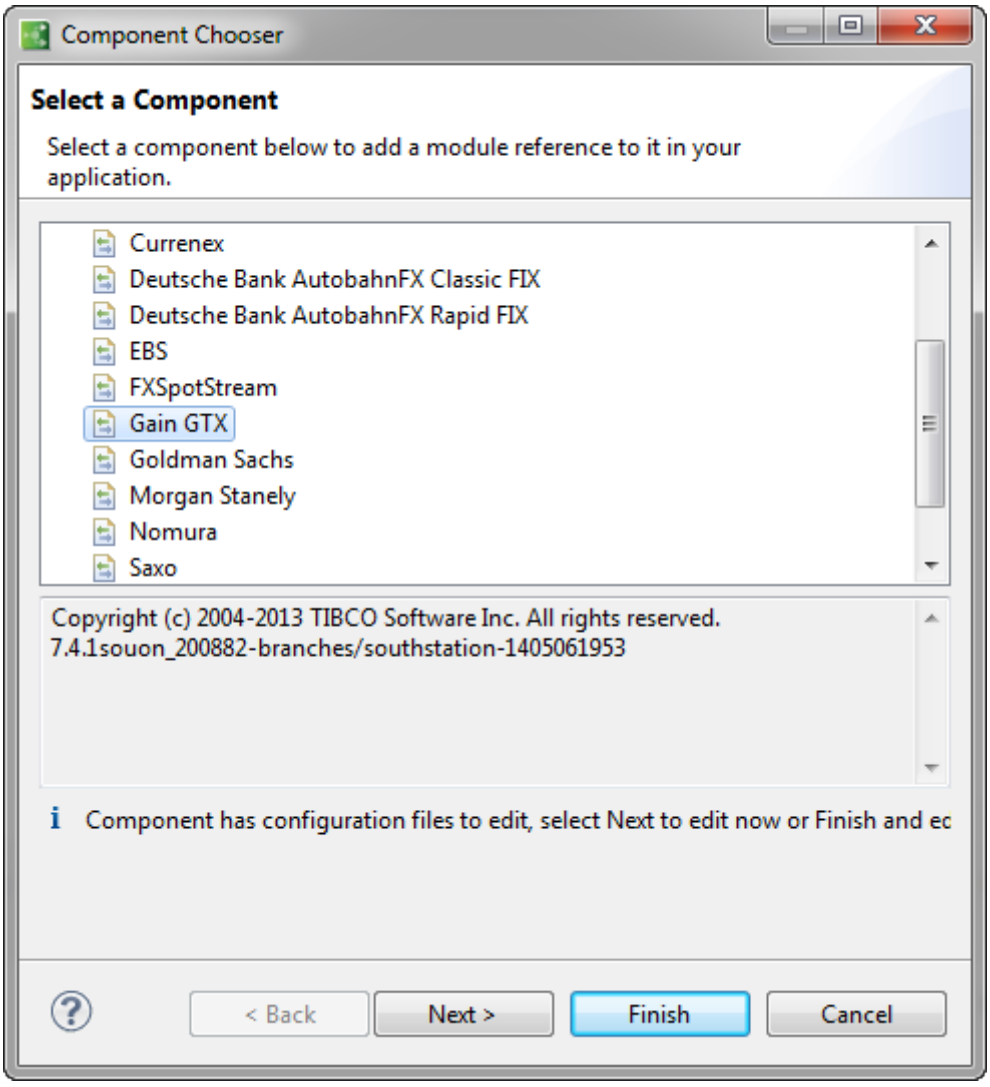
Accessing Trading Components

The Trading Components Framework is installed with StreamBase and is available in StreamBase Studio by default.

You load Market Data handlers and trade execution handlers separately for each venue you want to access in an application. The Trading Components palette tray has a Market Data Handler icon and an Execution Handler icon, as shown in this list view:

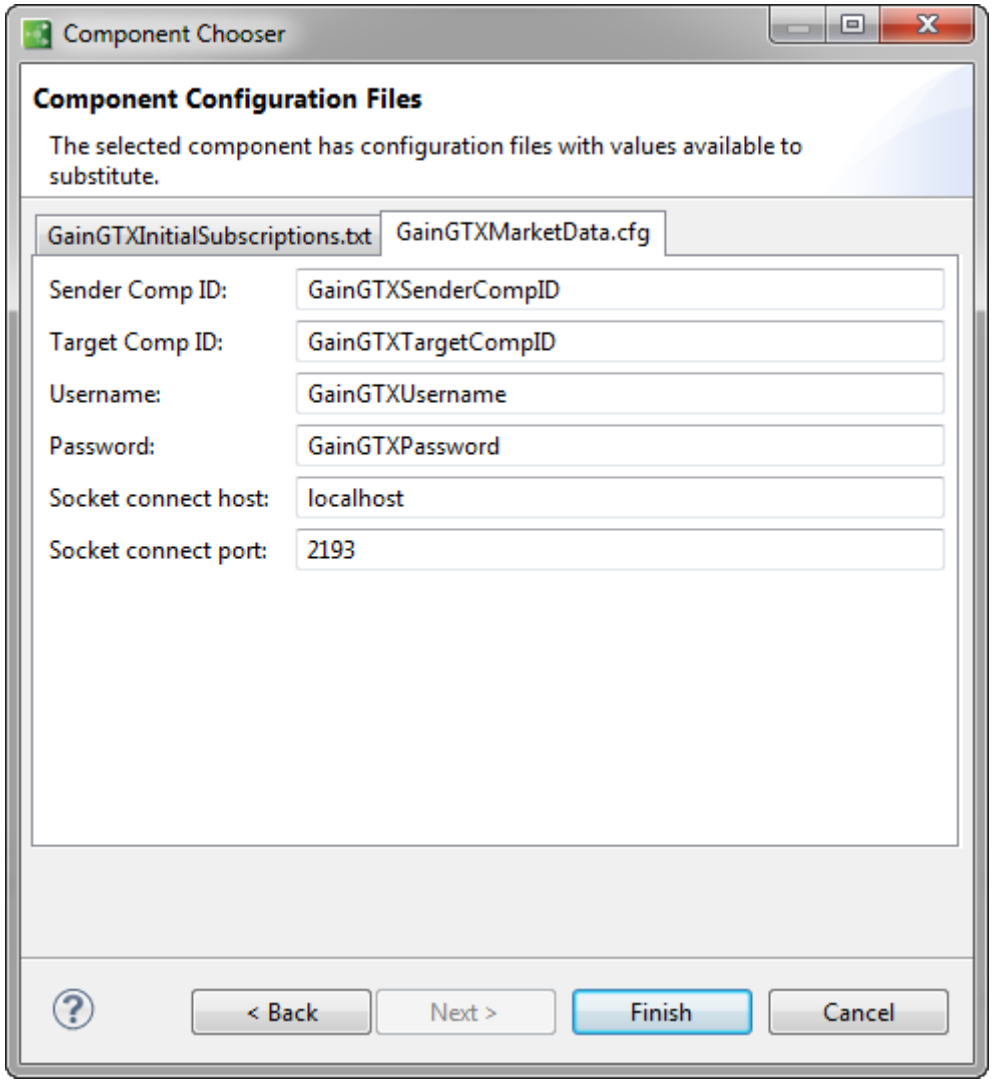


To add a handler for a venue to a StreamBase application, drag the appropriate icon to the canvas. A wizard opens for you to choose a handler, similar to the way you add a Java operator or adapter to a project. The following example shows the selection of a Gain GTX Market Data handler.

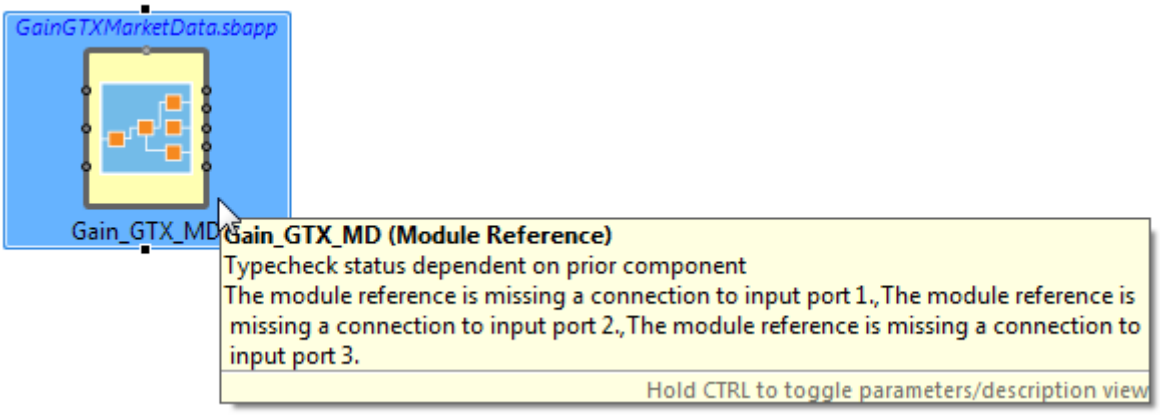


Clicking btn:[Finish] includes the selected handler with default configuration file entries and

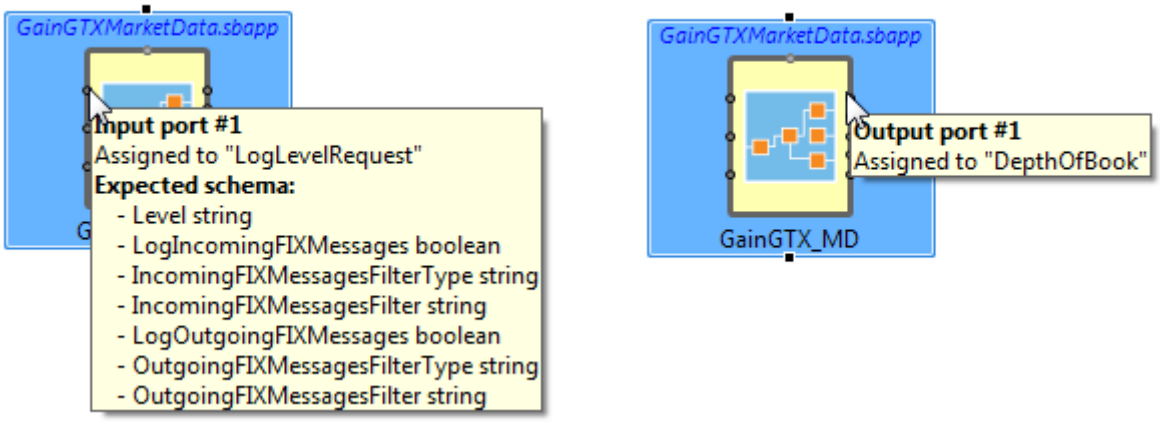
values. Clicking btn:[Next] allows you to edit the default configuration file name and key properties, as shown below for the Gain GTX Market Data handler.



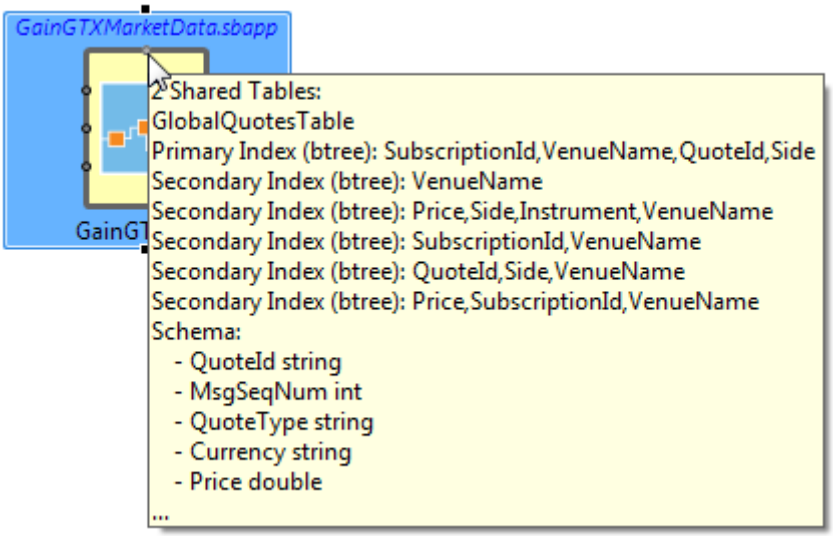
Clicking btn:[Finish] presents the module at the location you dragged it to with its label selected. It is good practice to provide a name in that field to identify this instance of the handler. Hovering over the module displays a tooltip describing required connections, as the following illustration shows.



Hovering over a port symbol displays the port number, its role, and its expected schema, as shown below for the first input and output ports of the Gain market data handler.



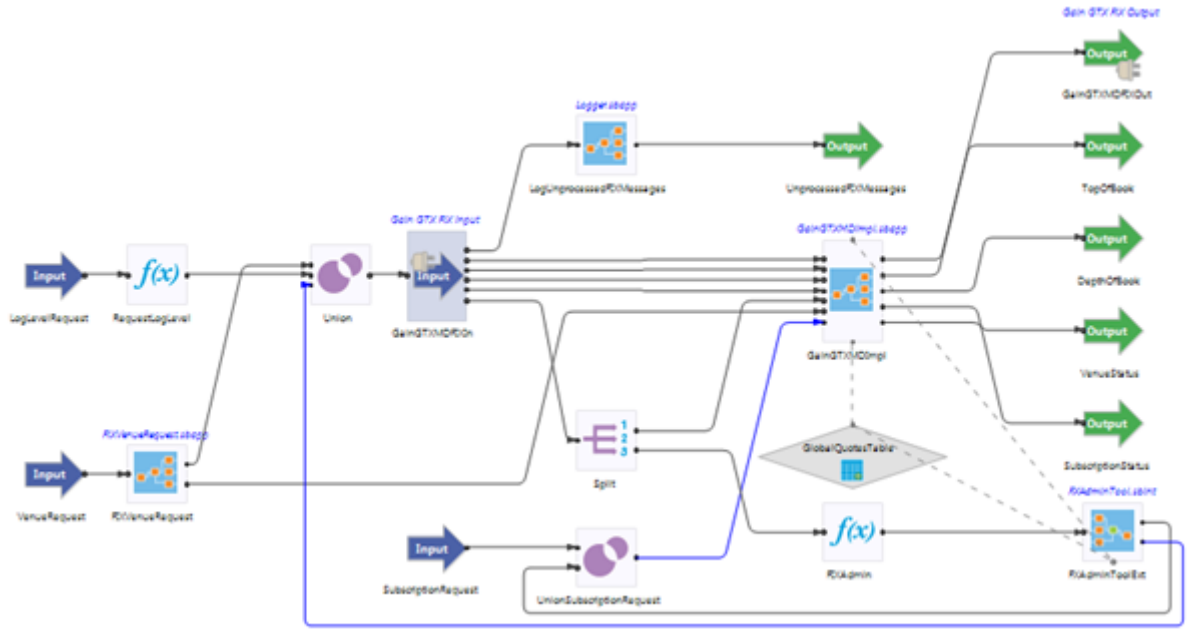
For handlers that reference shared tables, hovering on the connector along the top displays information about how tables are configured, as shown below.



NOTE Your application may not require that all ports be connected.

Each handler is a module containing components that do not require configuration or modification. For example, the Gain GTX Market Data handler shown above is implemented with four levels of modules. The top level module is shown below to illustrate.

IMPORTANT *You need not and should not modify the internals of Trading Component handlers.*



Learning from Trading Component Sample Applications

Trading Components provide uniform interfaces across all handlers. Samples are provided that illustrate how to configure them into a variety of applications. There are two samples for each Trading Component venue, one for accessing market data and another for executing trades.

Access the samples through menu:File[Load StreamBase Sample] , and then twist open the Trading Components section at the bottom of the Load StreamBase Samples dialog to see all the samples. Double-click one of the entries, or select it and click btn:[OK] . When you load any of the samples, StreamBase Studio automatically loads Trading Components if you have not done so previously.

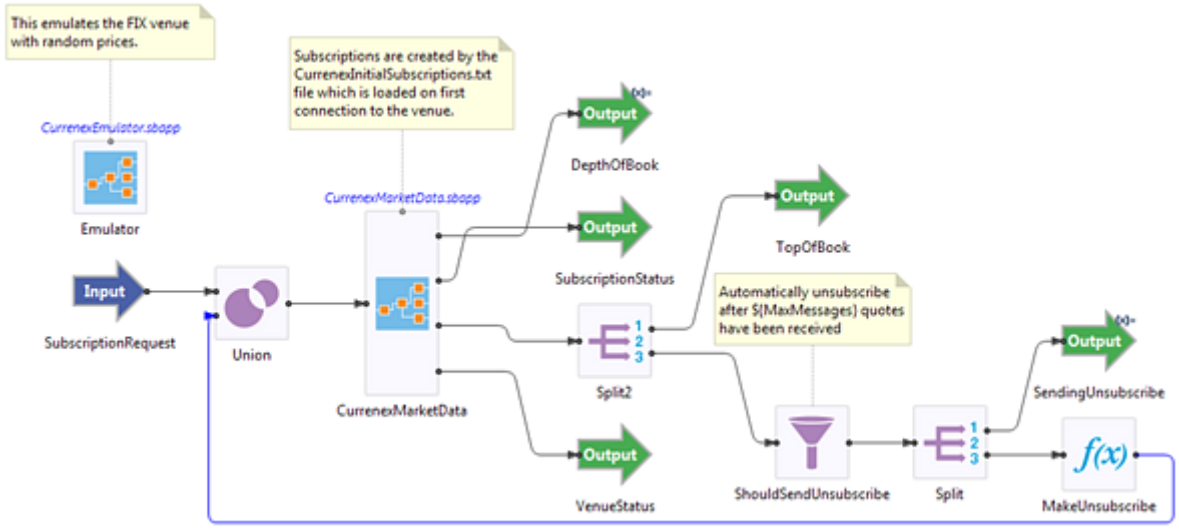
Many samples come with emulators that supply fictitious market data, and thus do not require connections to live data feeds to run. However, you do need to configure certain properties for most samples, such has credentials and socket connection.

The samples that include market data emulators are:

* * * * * * * *

The goes beyond the others to blend market data for two venues, illustrating how to configure multiple market data handlers to interleave quotes from different venues and then sort and query them as a group.

As an example, the following illustration shows the Currenex trading components market data sample application.



This application utilizes the prepackaged Trading Components module *CurrenexMarketData.sbapp* , and provides logic to subscribe to the (emulated) Currenex venue and then to unsubscribe after a set number of messages have been received (specified by the module parameter `${MaxMessages}`).

Supported Venues

A Trading Components venue is the source of a data feed. All supported venues are for FIX. Market Data handlers for the following venues are currently available in Trading Components. Nearly all venues also have execution handlers, as indicated in the second column.

NOTE

To connect to venues you must have purchased their associated premium adapters and in some cases downloaded them from tibco.com. Whether packaged with StreamBase software or separately, you are only entitled to use premium adapters that are listed in your contract. If the **Separate** column in the table contains Yes, the associated adapter comes as a separate download. For a complete list of standard and premium adapters and their usage restrictions, click here.

Venue	Execution	Streaming	RFQ Types	Folder name	Separate
Barclays BARX FIX	Yes	N/A	Spot, NDF	barclays-barx	No
Bloomberg Tradebook FIX	No	Spot	N/A	bbg-tradebook	Yes
CitiFX ESP	Yes	N/A	Spot, Forward, NDF	citifxesp	No
CitiFX Options	No	N/A	N/A	citifxoptions	No
Currenex Market Data	Yes	Spot, Forward	N/A	currenex	No

Deutsche Bank AutobahnFX Classic	Yes	Spot, Forward, Swap	N/A	db-classic-fix	No
Deutsche Bank AutobahnFX Rapid	Yes	Spot	N/A	db-rapid-fix	No
ICAP/EBS	Yes	Spot, NDF	N/A	ebs	No
FXSpotStream FIX	Yes	Spot, Forward	N/A	fxspotstream	No
GAIN GTX	Yes	Spot, Forward	N/A	gain-gtx	Yes
Goldman Sachs	Yes	Spot, Forward	Spot, Forward, NDF, Swap	gs	No
Morgan Stanley	Yes	N/A	Spot, Forward, NDF, Swap	ms	No
Nomura	Yes	Spot	Spot, Forward, Swap	nomura	No
Saxo Bank FIX	Yes	N/A	Spot	saxo	No
UBS Investment Bank	Yes	Spot, Forward, NDF	Spot, Forward, NDF, Swap	ubs	No

All venues behave similarly, with certain exceptions that are described in Notes on Specific Venues.

Subscription Validation

All venues validate subscriptions according to the following set of rules:

- If no instrument type is supplied then a default will be used, SPOT will be used for most venues and SPOT_RFQ will be used for those that SPOT is not valid.
- If the venue is down the subscription will fail.
- The venue will check against valid instrument types as listed in the venue list.
- If `currency1` or `currency2` does not have a value the subscription will be rejected.
- Tenor values will be checked per venue based on the current venue specification.
- Some venues will validate unsubscribe requests in the same manner as subscription requests.

Quote Tables

By default, quotes are held in tables on a per-venue basis. Such tables are not shared. However, your application can share market data for any venue with a *global quote table* , enabling it to aggregate, sort and compare quotes across venues. To share tables accessed by your application, on the module's Parameter tab add the parameter `GlobalBookTableType` and set it to `input`.

Notes on Specific Venues

This section provides additional descriptions of behaviors of certain venues that depart from the norm. For a complete list of venues, see Supported Venues.

- CitiFX Options (Volume)

  "Top/Depth of book" bid/ask use OptionsExtendedData

- Currenex
  - Market Data Request(V) always sends 'Full Book' req uest for market data (264=0) we do not currently support Top Of Book (264=1)
  - Market Data Request(V) always sends incremental refresh (265=1) no other options available for venue
  - Market Data Request(V) always sends AggregateBook request (266=Y) we do not currently support Non-Aggrgate
  - Market Data Request(V) supports requests for SIDE_BID/SIDE_ASK or both, these values are translated to Bid/Offer
  - RFQ (MsgType=R) is not currently supported.

- DB Rapid sub

  OutputMissingTiersAsZeroPriceItems parameter can be set to true to always output missing/old tier items as zero price items (MLP Request)

- ICAP/EBS

  Currently only FIX 5.0 venue which requires different logon

- FX Spot Stream
  - Bid/Ask ExtendedField1 gets value from MDEntryOriginator
  - OutputMissingTiersAsZeroPriceItems parameter can be set to true to always output missing/old tier items as zero price items (MLP Request)

Handler Ports, Parameters, Commands, and Constants

To configure a handler, you normally need to set certain module parameters. The tables on this page describe the parameters that can be set, the commands that can be issued, and literal constant values for parameters.

Parameters Common to All Venues

Beside their port parameters, all venues share the following parameters:

Parameter Name	Data Type	Description	Notes
VenueName	String	The name of the venue and handler, usually in the form `venue` + `handlerType`. For example, the `VenueName` for an execution handler is usually `venue` + "Execution", as in Deutsche Bank AutobahnFX Rapid FIX Execution.	
FIXHeartBeat	Integer	The interval between sending FIX heartbeat commands. This typically does not need to be changed, because it defaults to each venue's specification.	
QFJConfigFile	String	The QuickFIX/J configuration file used to connect to and from the venue. Normally defaults to `VenueName` + "MarketData" or "Execution" + *.cfg*	
SMTPHost	String	The host used when sending SMTP alerts when the venue goes up or down.	
SMTPPort	Integer	The port used when sending SMTP alerts when the venue goes up or down.	
SMTPFrom	String	The from e-mail address used when sending SMTP alerts when the venue goes up or down	

Parameter Name	Data Type	Description	Notes
SMTPTo	String	The send-to email address used when sending SMTP alerts when the venue goes up or down.	
SMTPSubject	String	The Subject used when sending SMTP alerts when the venue goes up or down.	
SMTPContentType	String	The content type used when sending SMTP alerts when the venue goes up or down. Default: text/html.	
SendSMTPAlerts	Boolean	A flag that specifies whether e-mails should be sent when the venue goes up or down.	
FIXEngineToUse	String	The FIX engine the adapter will use to connect to your FIX venues.	Default: "QuickFIX/J"/"StreamBase High-Performance Engine"
LogonWhenApplicationStart	String (`true` or `false`)	If `true`, the adapter immediately attempts to log on to the counterparty when the application starts.	
FieldNameMapFile	String	The exception to FIX field can be specified in this file.	
LogIncomingFIXMessages	String (`true` or `false`)	If `true`, the adapter enables logging of incoming FIX messages to the console. This property overrides a linked FIX input adapter's corresponding settings.	
IncomingMessageFilterType	String (`Include` or `Exclude`)	The type of filter logic for incoming FIX messages.	
IncomingMessageFilter	String	The filter used for incoming FIX messages.	

Parameter Name	Data Type	Description	Notes
LogOutgoingFIXMessages	String (`true` or `false`)	If `true`, the adapter enables logging of outgoing FIX messages to the console. This property overrides a linked FIX input adapter's corresponding settings.	
OutgoingMessageFilterType	String (`Include` or `Exclude`)	The type of filter logic for outgoing FIX messages.	
OutgoingMessageFilter	String	The filter used for outgoing FIX messages.	
ReportProfilingData	String (`true` or `false`)	If `true`, the adapter enables logging of FIX message average/min/max processing times to the console.	
ReportingInterval	Integer	The number of messages to log to the console.	
LogLevel	String	The verbosity to use for printing messages to the console, one of: OFF, ERROR, WARN, INFO, DEBUG, TRACE, or ALL.	
InitialSubscriptionsFile	String	The initial subscription file used for market data handler to subscribe to symbols as soon as a connection is made.	Market Data only
GlobalBookTableType	String (`concrete` or `input`)	Specifies whether the table declared in the MarketData.sbapp is a local table or a placeholder table. Specify `concrete` to share a quotes table across multiple venues, such as to enable Port Configurations aggregation and sorting for true top of book.	Market Data only

Parameters by Venue for Market Data (MD) and Execution (EX)

Some venues have additional parameters that you can and usually must specify. These are:

Venue	MD	EX	Parameter Name	Data Type	Default	Description
Barclays BARX			Account	String	YourAccount	The account to send with each trade request
CitiFX ESP			SubscriptionQuoteType	String	QUOTE_TYPE_RESTRICTED_TRADABLE	Determines the type of quote to received, options are: QUOTE_TYPE_INDICATIVE QUOTE_TYPE_TRADABLE QUOTE_TYPE_RESTRICTED_TRADABLE
CitiFX ESP			Account	String	YourAccount	The account to send with each market data or trade request
CitiFX ESP			ConvertZeroPriceOrSizeToNull	Boolean	true	If true then any quote that is received with a zero or null price/size will be converted to a entirely null quote.

Venue	MD	EX	Parameter Name	Data Type	Default	Description
CitiFX ESP			DefaultQuantity	Integer	1	If the Subscription.Quantity does not have a value then this default will be used for market data requests.
DBClassic			Account	String	YourAccount	The account to send with each market data or trade request
DBRapid			Account	String	YourAccount	The account to send with each market data or trade request
DBRapid			RequestPartialFills	Boolean	true	Used to determine TimeInForce sent to the venue: true = IOC (Immediate or cancel); false = FOK (Fill or Kill)
DBRapid			OutputMissingTiersAsZeroPriceItems	Boolean	false	Used to determine if missing tiers should be output as zero price items before being removed from the bid/ask list.

Venue	MD	EX	Parameter Name	Data Type	Default	Description
FXSpotStream			OutputMissingTiersAsZeroPriceItems	Boolean	false	Used to determine if missing tiers should be output as zero price items before being removed from the bid/ask list.
FXSpotStream			PartyIDsFile	String	FXSpotStreamPartyIDs.txt	The file containing the party Id's to be sent with each market data request. The file contains a single column of PartyIds with a header. See the FXSpotStream FIX Specification for valid party Ids .
FXSpotStream			Incremental Refresh	Boolean	true	Used to determine what type of market data updates to request: true - Incremental refresh; false - Full market data refresh

Venue	MD	EX	Parameter Name	Data Type	Default	Description
FXSpotStream			OutputIndividualMDEntryOriginators	Boolean	false	If set to true then each update (incremental or full) will produce multiple top of book and depth of book updates, one per Originator. (MDEntryOriginator is another term for PartyID)
Gain GTX			Account	String	YourAccount	The account to send with each market data or trade request
Goldman Sachs			Account	String	YourAccount	The account to send with each market data or trade request
Goldman Sachs			ConvertZeroPriceOrSizeToNull	Boolean	true	If true then any quote that is received with a zero or null price/size will be converted to a entirely null quote.

Venue	MD	EX	Parameter Name	Data Type	Default	Description
Goldman Sachs			DefaultQuantity	Integer	1	If the Subscription.Quantity does not have a value then this default will be used for market data requests.
Morgan Stanley			ConvertZeroPriceOrSizeToNull	Boolean	true	If true then any quote that is received with a zero or null price/size will be converted to a entirely null quote.
Morgan Stanley			Account	String	YourAccount	The account to send with each market data or trade request
Nomura			Account	String	YourAccount	The account to send with each market data or trade request
Saxo			Account	String	YourAccount	The account to send with each market data or trade request
UBS			UBSPartyID	String	SomeParty	Party Id (counterparty) for sending quote and trade requests.

Venue	MD	EX	Parameter Name	Data Type	Default	Description
UBS			UBSPartyRole	Integer	13	The type of party to send with trade requests. May be: 11 = Order Origination Trader; 13 = Order Origination Firm
UBS			ValidMarginMS	Integer	1000 ms	Specifies latency in ms for received quotes containing a new non-null ValidUntilTime. The system subtracts SendingTime and ValidMarginMS from ValidUntilTime and then sets that time as the new quote expiry time.
UBS			RFQExpirationMS	Integer	60000 ms	Time in milliseconds before the system considers this quote request expired and removes it.

Port Configurations

Trading component handles have three or more input and output ports connecting them to your

EventFlow application. The following sections describe these ports.

Market Data Handler Component Ports

Each Market Data Handler uses the same port configuration.

Input Ports

Market data input ports accept three streams. Click any stream name to see its schema definition:

Port	Module Stream	Description
1	LogLevelRequest	Configures which logging messages sent and received
2	SubscriptionRequest	Connects and disconnects from the current venue
3	VenueRequest	Connects to, Disconnects from, and requests status from a venue

Output Ports

Market data output ports issue up to five streams:

Port	**Module Stream**	**Description**
1	DepthOfBook	The pending orders for a security or currency
2	LogonResponse	Not always present
2 or 3	SubscriptionStatus	Whether the subscription is currently on, off, or pending.
3 or 4	TopOfBook	The best bid and ask prices
4 or 5	UnprocessedFIXMessages	Not always present
4 or 5	VenueConnectDisconnect	Not always present
4 or 5	VenueStatus	Current status of the venue's connection

Execution Handler Component Ports

Input Ports

Execution handler input ports accept three or four streams. Click the stream name to see its schema definition:

Port	Module Stream	Description
1	ExecutionRequest	The specifics of a trade
2	LogLevelRequest	How and what messages to log
3	SubscriptionRequest	Not always present. Connects, disconnects, and lists subscriptions
3 or 4	VenueRequest	Connects, disconnects, and gets venue status

Output Ports

Market data output ports issue two, three, or four streams:

Port	Module Stream	Description
1	Execution Report	Data describing how and when order was filled
2	UnprocessedFIXMessages	Not always present
2 or 3	SubscriptionRequest	Not always present
2, 3 or 4	VenueStatus	Current status of the venue's connection

Named Schemas

Much of Trading Components' ease of use derives from having a set of standardized schemas and constants that are used for all venues. These definitions are in the TC_fx_common project, which users should not attempt to modify. The following table summarizes the named schemas:

Named Schema from SharedSchemas.sbapp	Number of Fields	Description	Notes
AccountSchema	3	Identifies an account for the purposes of data reporting	
FIXAdminSchema	7	For communicating with the	Not exposed to venues
FIXInputAdapterCommandSchema	7		
FIXLogonResponseSchema	5		

Named Schema from SharedSchemas.sbapp	Number of Fields	Description	Notes
FIXTradingSessionStatusSchema	8	The fields from a FIX "Trading Session Status (h)" message that Trading Components uses	
LogLevelRequestSchema	7	Describes how to log and filter incoming and outgoing messages	
VenueFirstTimeUpSchema	2	Used to indicate when a connection first comes up	
VenueRequestSchema	1	A request to check the current status of a connection, connect to and disconnect from a venue	A VENUE_CMD_* constant
VenueSchema	2	A unique ID for a particular venue connection	venue name + connection name
VenueStatusSchema	6	The closed or open status (DOWN or UP) for a venue and other information as needed, generated on demand or asynchronously.	A VENUE_STATUS_* constant

The named schemas for passing venue trading data, such as instruments, quotes, execution requests, tenors and top and depth of book are defined in FXSharedSchemas.sbapp, and are summarized in the following table:

Named Schema from FXSharedSchemas.sbapp	Number of Fields	Description	Notes
FXDepthOfBookSchema	6	A schema that holds entire ladder of quotes. It is assumed that ladder is refreshed atomically in a single tick.	Contains list field and schema fields Instrument and Asks
FXExecutionReportSchema	14	A schema to identify an update to information about an execution request	

Named Schema from FXSharedSchemas.sbapp	Number of Fields	Description	Notes
FXExecutionRequestSchema	22	A schema to identify a request for an instrument to be bought or sold	Contains schema field User
FXGlobalQuoteDataSchema	22	Used to populate global quote tables	Contains schema fields Instrument and OptionsExtendedData
FXInitalSubscriptionSchema	13	Like FXSubscriptionSchema but with the Instrument fields flattened and without the SubscriptionId field. Used as the schema of the "initial subscriptions" file.	
FXInstrumentSchema	7	Fully identifies kind of item to be bought and sold.	
FXInvalidSubscriptionRequestSchema	2	Wraps FXSubscriptionSchema for requests that are rejected	Contains schema field Subscription
FXInvalidSubscriptionRequestWithStatusSchema	3	Wraps FXInvalidSubscriptionRequestSchema to provide status information	Contains schema field Subscription
FXQuoteDataOptionsExtendedDataSchema	12	Extends FXQuoteDataSchema	
FXQuoteDataSchema	18	Quote data includes price, quantity, and other information as needed.	Contains schema field
FXSubscriptionInfoSchema	5	Like FXSubscriptionSchema but without the SubscriptionId field; for use as the schema of the "initial subscriptions" file	Contains schema field Instrument

Named Schema from FXSharedSchemas.sbapp	Number of Fields	Description	Notes
FXSubscriptionListSchema	11	A data structure used to manage a subscription to a particular instrument	Contains schema field Instrument
FXSubscriptionRequestSchema	2	A request to subscribe to/unsubscribe from an instrument or describe/list subscriptions (see FXSubscriptionStatusSchema)	Contains schema field Subscription
FXSubscriptionSchema	8	A data structure used to manage a subscription to a particular instrument	Contains schema field Instrument
FXSubscriptionStatusSchema	3	A response to a request (see FXSubscriptionRequestSchema) to sub/unsub, describe a subscription, or an async event about a subscription	Contains schema field Subscription
FXTenorSchema	2	The representation of a tenor in structured form.	
FXTopOfBookSchema	6	A data structure to hold the top bid and ask quotes	Contains schema fields Instrument, Bid, and Ask

The table schemas for holding venue trading data, such as venue names, timestamps, prices, quantities and more are also defined in FXSharedSchemas.sbapp. They are:

Table Schema from FXSharedSchemas.sbapp	Number of Fields	Description	Notes
FXGlobalQuotesTableSchema	22	A table schema that holds entire ladder of quotes for multiple venues. Uses FXGlobalQuoteDataSchema.	Contains list field and schema fields Instrument and Asks

Table Schema from FXSharedSchemas.sbapp	Number of Fields	Description	Notes
FXQuotesTableSchema	6	A table schema that holds entire ladder of quotes for a single venue. Uses FXQuoteDataSchema.	
FXSubscriptionListTableSchema	11	A table to manage a subscriptions to instruments. Uses FXSubscriptionListSchema.	Contains schema field User

Log Level Request Fields

You can change the logging level for each venue at any time by sending commands to its LogLevelRequest input stream. That stream lets you change both the basic log level and the level for incoming and outgoing FIX messages. The schema for this request is:

Field	Data Type	Description
Level	String	The log level to set for the FIX adapters using the normal adapter log levels (OFF, ERROR, WARN, INFO, DEBUG, TRACE, ALL)
LogIncomingFIXMessages	Boolean	If enabled, the FIX adapter will start logging formatted incoming FIX messages.
IncomingFIXMessagesFilterType	String	Specifies how to filter incoming FIX messages. Set to either 'Include' or 'Exclude'
IncomingFIXMessagesFilter	String	A comma-separated list of message types (that is, the value of FIX tag 35) on which to filter incoming messages
LogOutgoingFIXMessages	Boolean	If enabled, the FIX adapter will start logging formatted outgoing FIX messages.
OutgoingFIXMessagesFilterType	String	Specifies how to filter outgoing FIX messages. Set to either 'Include' or 'Exclude'

Field	Data Type	Description
OutgoingFIXMessagesFilter	String	A comma-separated list of message types (that is, the value of FIX tag 35) on which to filter outgoing messages

Notice that you cannot both include and exclude messages using a filter.

Venue Request and Status Commands

The VenueRequest stream enables the end user to turn a venue connection on and off or request its status at any time, the status will be outputted from the VenueStatus output stream:

Field	Type	Description
Command	string	One of the Venue Commands listed below

The VenueRequest input stream recognizes the following constant venue commands in its Command field:

Constant	Value	Description
VENUE_CMD_CONNECT	CONNECT	Request the current connection connect to the venue
VENUE_CMD_DISCONNECT	DISCONNECT	Request the current connection disconnect from the venue
VENUE_CMD_STATUS	STATUS	Request the current connection status of the venue, outputted on the VenueStatus stream

Venue status is described by the following constants:

Constant	Value	Description
VENUE_STATUS_UP	UP	Indicates that currently connected venue is online
VENUE_STATUS_DOWN	DOWN	Indicates that currently connected venue is offline

Market Data Fields

You subscribe to a venue by sending it a request with the following data fields. This is the FXSubscriptionRequestSchema, defined in TC_fx_handlers/common/FXSharedScemas.sbapp:

Field	Type	Description
Command	String	One of the SUB_CMD_* values defined in the table below.
Subscription.SubscriptionId	String	The ID of the subscription. If left blank, an ID will be generated.
Subscription.Side	String	One of the SIDE_* constants
Subscription.MarketDepth	Integer	All available, if null
Subscription.Quantity	Double	If set and supported by the venue, produces a stream of prices for a single volume for the spot, forward or near leg of a swap
Subscription.QuantityFar	Double	If set and supported by the venue, produces a stream of prices for a single volume for the far leg of a swap (null ⇒ far leg is the same as near leg)
Subscription.Currency	String	The trading currency
Subscription.Duration	Integer	Duration in seconds of stream quotes. Not implemented by all venues.
Subscription.Instrument.InstrumentType	String	One of the INST_TYPE_* constants defined below
Subscription.Instrument.Currency1	String	The first currency involved in an FX trade
Subscription.Instrument.Currency2	String	The second currency involved in an FX trade
Subscription.Instrument.SettlementDate	Timestamp	The settlement or value date of the spot, forward or near leg of the swap
Subscription.Instrument.SettlementDateFar	Timestamp	The settlement date of the far leg of the swap. Null if not a swap.
Subscription.Instrument.Tenor	String	Tenor of a forward or tenor of the near leg of a swap. Null if not a swap or a broken date. Use one of the TENOR_* constants

Field	Type	Description
Subscription.Instrument.TenorFar	String	Tenor of the far leg of a swap. Null if not a swap or a broken date. Use one of the TENOR_* constants

Market Data Constants

* SubscriptionRequest streams* recognize the following constant venue commands in its Command field:

Constant	Value	Description
SUB_CMD_SUBSCRIBE	SUBSCRIBE	Request the current connection connect to the venue.
SUB_CMD_UNSUBSCRIBE	UNSUBSCRIBE	Request the current connection disconnect from the venue
SUB_CMD_LIST	LIST	Output all current subscriptions (currently not implemented)
SUB_CMD_DESCRIBE	DESCRIBE	Output subscription information (currently not implemented)

The current status of a subscription is indicated by one of the following set of constant values defined in TC_Common/common/SharedSchemas.sbapp:

Constant	Value	Description
SUB_STATUS_UP	UP	The subscription is now active.
SUB_STATUS_DOWN	DOWN	The subscription is no longer active.
SUB_STATUS_SENT	SENT	The subscription has been sent to the venue but is not yet active.

Instrument Types are constants defined in TC_Common/common/SharedSchemas.sbapp. They are:

Constant	Value	Description
INST_TYPE_SPOT	SPOT	A spot market data request; to perform a RFQ request SPOT_RFQ
INST_TYPE_SWAP	SWAP	A spot market data request; to perform a RFQ request SWAP_RFQ

Constant	Value	Description
INST_TYPE_NDF	NDF	A spot market data request; to perform a RFQ request NDF_RFQ
INST_TYPE_MLEG	MLEG	A spot market data request; to perform a RFQ request MLEG_RFQ
INST_TYPE_FUT	FUT	A spot market data request; to perform a RFQ request FUT_RFQ
INST_TYPE_FO	FO	A spot market data request; to perform a RFQ request FO_RFQ. Only the following venues allow RFQ requests: Barclays BARX, Goldman Sachs, Morgan Stanley, Nomura, and UBS
INST_TYPE_RFQ_SUFFIX	_RFQ	Append this suffix to the end of any of the above values to create a request for quote. RFQ quote requests are usually similar to market data request but are used on venues that allow for trades to be done with type QUOTED, in which you trade using a *quoteId* .

Pricing Types are constants defined in TC_Common/common/SharedSchemas.sbapp. They are:

Constant	Value	Description
PRICING_TYPE_BOOK	BOOK	A standard quote type
PRICING_TYPE_RFQ	RFQ	The quote is produced by a RFQ request.

Quote Types are constants defined in TC_Common/common/SharedSchemas.sbapp. They are:

Constant	Value	Description
QUOTE_TYPE_INDICATIVE	INDICATIVE	The quote is provided by a market maker to a trading party but that is not firm.
QUOTE_TYPE_INVALID	INVALID	The quote is invalid.
QUOTE_TYPE_RESTRICTED_TRADABLE	RESTRICTED_TRADABLE	The quote provided is restricted tradable usually by date or time span.

Constant	Value	Description
QUOTE_TYPE_TRADABLE	TRADABLE	The quote provided is tradable on the market.

Side Values are constants defined in TC_Common/common/SharedSchemas.sbapp. They are:

Constant	Value	Description
SIDE_ASK	ASK	Used by some the market data handlers to request the ask side of a quote.
SIDE_BID	BID	Used by some the market data handlers to request the sell side of a quote.
SIDE_BOTH	BOTH	Used by some the market data handlers to request both the ask side and the sell side of a quote.
SIDE_BUY	BUY	Used in the execution handlers to determine the bid/buy side
SIDE_SELL	SELL	Used in the execution handlers to determine the bid/sell side

Tenor Values are constants defined in TC_Common/common/SharedSchemas.sbapp. The values below are used internally for making subscriptions requests. As each venue has its own set of valid tenors, these values will be translated to a ones appropriate to the venue.

Constant	Value	Description
TENOR_BUSINESS_DAYS	B	When used, B always has a number appended. Some valid values are: B1, B2, B40. However, currently these values are only supported for the Goldman Sachs venue.
TENOR_DAYS	D	When used, D always has a number appended. Some valid values are: D1, D2, D40
TENOR_IMM	I	When used, I always has a number appended. Some valid values are: I1, I2, I3
TENOR_MONTHS	M	When used, M always has a number appended. Some valid values are: M1, M2, M6
TENOR_NEXT_DAY	ND	
TENOR_OVERNIGHT	ON	

Constant	Value	Description
TENOR_SPOT	SP	
TENOR_SPOT_NEXT	SN	
TENOR_TODAY	TOD	TOD is sometimes also allowed to have a numeric value depending on the venue.
TENOR_TOMORROW	TOM	
TENOR_TOMORROW_NEXT	TN	
TENOR_WEEKS	W	When used, W always has a number appended. Some valid values are: W1, W2, M10
TENOR_YEARS	Y	When used, Y always has a number appended. Some valid values are: Y1, Y2, Y5

Trade Execution Fields

Execution requests use the following FXExecutionRequestSchema schema, defined in defined in TC_Common/common/SharedSchemas.sbapp.

Field	Type	Description
Command	String	One of the EXEC_CMD_* constant values
RequestId	String	The client-provided ID of this request, which must be supplied
OrderType	String	One of the ORD_TYPE_* constants
QuoteId	String	Used when a venue requires that specific QuoteId it issued accompany an execution request
SubscriptionId	String	The subscriptionId of the tick that the quote was obtained from
Side	String	One of the SIDE_TYPE_* constants
Quantity	Double	The quantity of the order in units of the specified denomination Currency

Field	Type	Description
QuantityFar	Double	The swap far leg amount
Currency	Double	The currency in which Quantity is specified
Price	Double	The rate expressed as Currency2 per Currency1
PriceFar	Double	The all-in far leg swap rate
StopPrice	Double	Used when Order Type is STOP or STOP_LIMIT
TimeInForce	String	Use one of the TIF_* constants
Expiration	Timestamp	When TimeInForce is GTD, the order will expire and be canceled if not filled by the time given. Partial fills may have occurred.
OriginalRequestId	String	Needed only for ORD_TYPE_CANCEL and ORD_TYPE_REPLACE; the client-provided ID that was used in the original request which is now being cancelled or replaced.
OriginalOrderId	String	Analogous to OriginalRequestId, except the venue-assigned ID for the request; can always be set to null on input to OrderStateManager
ParentRequestId	String	The request ID of the "parent" request that this request is to be associated with; needed only when requests are being passed through the parent/child order tracker. Currently not implemented in any Venue.
StrategyAlgo	String	The name/ID of the execution strategy that should be used to execute this request and generally ignored by Framework code. Applications can use this field to hold strategy routing information
RequestTime	Timestamp	The timestamp of this request

Field	Type	Description
VenueName	String	Used to router the request to a particular execution handler; unused by the handler itself.
User	AccountSchema	Not implemented.
Instrument.InstrumentType	String	Use one of the INST_TYPE_* constants
Instrument.Currency1	String	The first currency involved in an FX trade
Instrument.Currency2	String	The second currency involved in an FX trade
Instrument.SettlementDate	Timestamp	The settlement or value date of the spot, forward or near leg of the swap
Instrument.SettlementDateFar	Timestamp	The settlement date of the far leg of the swap. Null if not a swap.
Instrument.Tenor	String	Tenor of a forward or tenor of the near leg of a swap. Null if not a swap or a broken date. Use one of the TENOR_* constants.
Instrument.TenorFar	String	Tenor of the far leg of a swap. Null if not a swap or a broken date. Use one of the TENOR_* constants.

Trade Execution Constants

* Execution Commands* are constants defined in TC_Common/common/SharedSchemas.sbapp. They are:

Constant	Value	Description
EXCE_CMD_CANCEL	CANCEL	Cancel an order
EXCE_CMD_NEW	NEW	Place a new order
EXCE_CMD_REPLACE	REPLACE	Replace current order
EXCE_CMD_STATUS	STATUS	Report status of orders

Order types are constants defined in TC_Common/common/SharedSchemas.sbapp. They are:

Constant	Value	Description
ORD_TYPE_STOP	STOP	Send a stop order
ORD_TYPE_STOP_LIMIT	STOPLIMIT	Send a stop limit order
ORD_TYPE_PREV_QUOTED	QUOTED	Send a previously quoted order; this order type is almost always used in conjunction with RFQ requests which provide a QuoteId to send along with the order.
ORD_TYPE_MARKET	MARKET	Send a market order
ORD_TYPE_LIMIT	LIMIT	Send a limit order
ORD_TYEP_AVERAGE_LIMIT	AVERAGE_LIMIT	Send an average limit order

Order Status strings are constants defined in TC_Common/common/SharedSchemas.sbapp. They are:

Constant	Value	Description
ORD_STATUS_ACCEPT	ACCEPT	The order has been accepted
ORD_STATUS_CANCEL	CANCEL	The order has been cancelled
ORD_STATUS_COMPLETE	COMPLETE	The order is completed
ORD_STATUS_DONEFORDAY	DONEFORDAY	The order is done for the day, usually associated with GFD time in force.
ORD_STATUS_ERROR	ERROR	The order has an error
ORD_STATUS_EXPIRE	EXPIRE	The order expired, usually associated with some time in force which would expire.
ORD_STATUS_PARTIAL	PARTIAL	The order partially executed
ORD_STATUS_PENDING	PENDING	The order is pending but not accepted
ORD_STATUS_REJECT	REJECT	The order was rejected
ORD_STATUS_REPLACE	REPLACE	The order was rejected

Time In Force types are constants defined in TC_Common/common/SharedSchemas.sbapp. They are:

Constant	Value	Description
TIF_OPG	OPG	Use OPG to send a market-on-open (MOO) or limit-on-open (LOO) order.

Constant	Value	Description
TIF_MARKET_ON_OPEN	MARKET_ON_OPEN	Market on open
TIF_MARKET_ON_CLOSE	MARKET_ON_CLOSE	Market on close
TIF_IOC	IOC	Immediate or Cancel
TIF_GTD	GTD	Good until date/time which must be specified
TIF_GTC	GTC	Good until cancel
TIF_FOK	FOK	Fill or kill
TIF_EXTENDED_DAY	EXTENDED_DAY	Extended day
TIF_DAY	DAY	Day