# ActiveX API Guide

Version 12.7.1

Sterling Trading Tech offers advanced API development integration as part of its Sterling Trader Pro platform. In order to work with the developers, we have provided an API guide, examples and technical support. We are continuously working to develop and advance our API functionality along with our front-end platform. In this guide you will find some basic examples of code to work with our API along with the fundamental library of components. This guide is primarily geared toward development in VB. Development in other languages is possible but is not supported as well by the API interface. Within the guide we will refer to functions as methods and members as properties. More properties and objects exist in the type library than you do in this guide mostly due to backward compatibility issues.

In addition to this guide developers have access to our online examples and our support team. Questions concerning the API can be directed to support@sterlingtradingtech.com or 312-346-9600 x 290. If you wish to be added to our developers email list, please send a request to support@sterlingtradingtech.com with your email address and the subject, “Add to STI API Developers List”.

## Table of Contents

Referencing the Type Library ..... 2
The Visual Basic 6.0 Development Environment ..... 2
Development in Alternative Environments ..... 3
Sterling ActiveX API Performance Considerations ..... 4
Visual Basic Coding Examples ..... 4
Visual C# Coding Examples ..... 5
Tracking Orders Using a Client Order ID ..... 7
Handling Rejections in the API ..... 8
Backward Compatibility and Algos ..... 8
Sterling ActiveX Object Library ..... 9
STIApp ..... 16

## Referencing the Type Library

ActiveX components can contain numerous classes, each with one or more programming interfaces. These interfaces can have methods and properties. The components can also have many enumeration constants which are symbolic names for constants that are passed or returned over the interface. For name scoping and management, each application defines its own group of these definitions into a type library. The type library is used by programming languages to check the correctness of calls to the component and it is used by COM when it creates the data packet which conveys a method call from one Windows process to another. (This data packet creation is called marshalling.)
You must reference the Sterling Type Library to communicate with the Sterling Trader Pro System. This type library gets installed when you install the Sterling Trader Pro. Before you can write Visual Basic code to communicate with the Sterling System you must reference the type library from within your Visual Basic project.

To reference these type libraries from Visual Basic, do the following:

1. From the Visual Basic Menu Bar, find the References menu item. Depending on the version of Visual Basic that you are running, this is either under the Project (VB6) or under the Tools (VBA) menu.
2. Select this menu item to bring up the references dialog. This dialog lists every registered type library on the System. This list is divided into two groups. The type libraries which are already referenced by the project are listed first. All of the remaining registered system type libraries are listed after that in alphabetical order.
3. Find the type library labeled "Sterling 1.0 Type Library" and click the check box to add the reference to your project. (Simply selecting the line is not sufficient; you have to check the box.)
4. Click OK to activate these changes.

After referencing this library, you can use it immediately. Also, if you reopen the references dialog, you will see that this library has been moved up to the top of the list with the other referenced libraries.

Each type library has a name, known as the Library Name, that is used in programming to qualify (or scope) all names used within it (such as components, methods, constants, and so on). Take note, however, that this name is not necessarily identical to the name of the file containing the type library. Visual Basic will look up the names in your program by going through the referenced type libraries in the order that they are listed in the references dialog. If two type libraries contain the same name, then Visual Basic will get its definition from the first library in the list, unless the name is qualified with a library name.
The Sterling library name is "SterlingLib". You can use this to qualify any identifier defined in that library.

## The Visual Basic 6.0 Development Environment

After the type library is referenced, its definitions become available to the Visual Basic Object Browser. From the Visual Basic development environment, the Object Browser is typically available via a toolbar icon, a menu selection or the F2 key.

The Object Browser can show all referenced type libraries at one time or can focus on any one of them. To focus on the base Sterling Type Library, select "SterlingLib" from the drop-down box in the upper left. The left pane of the Object Browser will then show only the component classes, interfaces and enumerations for SterlingLib. If you select a component (such as STIOrder) on the left, its methods and

properties will be shown on the right. When you select an item in the right pane, more complete information (such as the parameters to pass to a method and a brief description of the method) are shown at the bottom.

Referencing a type library also activates Visual Basic Intellisense™ for all programming language names in the library. For instance, as you begin to type in a name like SterlingLib.STIOrder, you will see the Visual Basic editor display a pop-up list of possible names to complete the typing. As you code a method call, Visual Basic will show you each parameter that you need to provide.

## Development in Alternative Environments

Sterling allows for the development in languages other than VB 6.0. This however does come with some setbacks. For example, development in .net based languages will cause a delay in data transmission due to the inefficiencies in communicating between .net and the ActiveX events. We have dealt with this inefficiency by adding the XML functions to our API. They allow you to receive events more quickly. Depending on your VS Version, you may need to open your project by right clicking the program and then selecting Run as administrator.

Using XML:

In order to receive the XML events, you will need to first set the mode to XML. To do this you will go under STIAPP on page 16 in the guide and set SetModeXML to true. Then you will need to choose the proper event. One such XML event is OnSTIQuoteUpdateXML vs OnSTIQuoteUpdate. You will also need to check for the XMLSerializer file with IO.File.Exists. Finally, you will need to decode the XML.

Decoding the XML:

In C#:

```
Private void OnSTIQuoteUpdateXML(ref string strQuote)
        XmlSerializer xs = new
        XmlSerializer(typeof(sSterlingLib.structSTIQuoteUpdate)); Sterling
        Lib.structSTIQuoteUpdate sstructQuote =
                (SterlingLib.tructSTIQuoteUpdate)xs.Deserialize(new
        stringReader(strQuote)); In VB.NET
Private Sub OnSTIQuoteUpdateXML(ByRef bstrQuote As String)
        Dim xs As New
        XmlSerializer(GetType(SterlingLib.tructSTIQuoteUpdate) Dim sr
        As New StringReader(bstrQuote)
        Dim structQuote As SterlingLib.structSTIQuoteUpdate = DirectCast(xs.Deseriialize(sr),
SterlingLib.structSTIQuoteUpdate)
        sr.Close()
```

The members of structQuote can now be accessed.

## Sterling ActiveX API Performance Considerations

The ActiveX API delay has been reduced to produce a maximum of 20 orders per second.

Cancel requests can now be sent for any open orders which have not had any activity within the past ten seconds regardless of their current state; i.e. – no further order confirm messages are being received from the exchange. Therefore, the order state says “Pending” – the order appears to be “frozen”. Such an order can now be canceled after 10 seconds, either from within the ActiveX API itself or from the Trading Monitor screen.

## Visual Basic Coding Examples

**Create a reference to the Sterling ActiveX Library**
- Select Project • References… from the menu.
- Select Sterling 1.0 Type Library from the Available References.
- If the Sterling 1.0 Type Library is not found, select Browse and find the Sterling.tlb file.

**Enable Events**
- Place the following line in the general section of your code to declare the object:

```
Dim WithEvents m_STIEvents As STIEvents
```

- Place the following line in an initializing procedure, such as the Form_Load procedure:

```
Set m_STIEvents = New STIEvents
```

- Now select m_STIEvents in the Object drop down control in your code window (top left drop down control). You will see the available events in the Procedure drop down control (top right drop down control). Select the event that you would like to catch, and it will be inserted into your code. It should look something like this:

```
Private Sub m_STIEvents_OnSTIOrderUpdateMsg(ByVal oSTIOrderUpdateMsg As
ISTIOrderUpdateMsg)
```

- Use the oSTIOrderUpdateMsg object to gather the information from the message.

**Sending an Order**
- Create the order object with the following code:

```
Dim order As StiOrder
Set order = New STIOrder Dim storder As structSTIOrder
```

- Fill the oroder properties with order information

```
storder.Account = “ACCT7” storder.Side = “B”
storder.Symbol = “CSCO” storder.Quantity = 500 storder.PriceType = ptSTIMkt storder.Tif = “D”
storder.Destination. = “NYSE”
```

- Create CLOrderId:

```
Dim theTime As SYSTEMTIME
GetLocalTime theTime
Storder.bstrClOrderID = storder.Account & ostheTime.wTime.wMonth & theTime.wDay &
theTime.wHour & theTime.wSecond & theTime.wMilliseconds
```

- Check for Errors

```
Text1.Text = orderSubmitOrderStruct(os)
```

**Canceling an Order**
Place the following line in the general section of your code to declare the object:

```
Dim m_STIOrderMaint As STIOrderMaint
```

- Place the following line in an initializing procedure, such as the Form_Load procedure:

```
Set m_STIOrderMaint = New STIOrderMaint
```

- Call the CancelOrder Method of the STIOrderMaint object for passing the required order information.

You can use either the OrderRecordID or the ClOrdID to cancel an order.
- OrderRecordID is the value that you get back in the OrderUpdateMsg. This is a unique order ID generated by the Sterling Trader® System to track an order.
- OldClOrdID is the client-generated order ID passed-in when the order is first sent.

- ClOrdID is an optional field. It is the ActiveX API client-generated order ID used for canceling the order record. It must be a unique ID, and it must remain unique over multiple trading days.

- To cancel with the Client Order ID:

```
m_STIOrderMaint.CancelOrder “<Account>”, 0, “<Client Order Id of Order to cancel>”,
“<Client
```

Order Id of new cancel order record>”

- To cancel with the Record ID:

```
m_STIOrderMaint.CancelOrder “<Account>”, <Record ID>, “”, “”
```

## Visual C# Coding Examples

**Create a reference to the Sterling ActiveX Library**
- Select Project • Add Reference… from the menu.
- Select Sterling 1.0 Type Library from the COM objects category.
- If the Sterling 1.0 Type Library is not found, select Browse and find the Sterling.tlb file.

**Enable Events**
- Place the following lines in your class declaration to declare the objects:

```
private SterlingLib.STIEvents m_STIEvents;
private SterlingLib.STIApp m_STIApp;
```

- Place the following lines in an initializing function, such as the Form constructor:

```
m_STIEvents = new SterlingLib.STIEvents();
m_STIApp = new SterlingLib.STIApp(); m_STIApp.SetModeXML(true);
```

- Now assign a handler for OrderUpdate event (XML parsing example is given on the page 5):

```
stiEvents.OnSTIOrderUpdateXML += new
SterlingLib._ISTIEventsEvents_OnSTIOrderUpdateXMLE
ventHandler(StiEvent_OnSTIOrderUpdateXML);
```

- Use the SterlingLib.structSTIOrderUpdate object to gather the information from the message.

**Sending an Order**

- Create the order object with the following code:

```
SterlingLib.STIOrder order = new SterlingLib.STIOrder();
SterlingLib.structSTIOrder storder = new SterlingLib.structSTIOrder();
```

- Fill the order properties with order information:

```
storder.bstrAccount = “ACCT7”; storder.bstrSide = “B”;
storder.bstrSymbol = “CSCO”;
storder.nQuantity = 500;
storder.nPriceType = SterlingLib.STIPriceTypes.ptSTIMkt;
storder.bstrTif = “D”;
storder.bstrDestination = “NYSE”;
```

- Create ClOrderId:

```
storder.bstrClOrderId = “ACCT7”
+ DateTime.Now.Year
+ DateTime.Now.Month
+ DateTime.Now.Day
+ DateTime.Now.Hour
+ DateTime.Now.Minute
+ DateTime.Now.Second
+ DateTime.Now.Millisecond;
```

- Submit order and check for errors:

```
int res = order.SubmitOrderStruct(storder);
```

**Canceling an Order**

- Place the following lines in your class declaration to declare the objects:

```
private SterlingLib.STIOrderMaint m_STIOrderMaint;
```

- Place the following lines in an initializing function, such as the Form constructor:

```
m_STIOrderMaint = new SterlingLib.STIOrderMaint();
```

- Call the CancelOrder Method of the STIOrderMaint object for passing the required order information.

You can use either the OrderRecordID or the ClOrdID to cancel an order.

- OrderRecordID is the value that you get back in the OrderUpdateMsg. This is a unique order ID generated by the Sterling Trader® System to track an order.
- OldClOrdID is the client-generated order ID passed-in when the order is first sent.

- ClOrdID is an optional field. It is the ActiveX API client-generated order ID used for canceling the order record. It must be a unique ID, and it must remain unique over multiple trading days.

To cancel with the Client Order ID:

m_STIOrderMaint.CancelOrder("<Account>", 0, "<Client Order Id of Order to cancel>", "<ClientOrder Id of new cancel order record>");

To cancel with the Record ID:

m_STIOrderMaint.CancelOrder("<Account>", <Record ID>, "", "");

## Tracking Orders Using a Client Order ID

The Client Order ID is used for the purpose of assigning an ActiveX API client-generated ID to an order *before* that order is initially sent. The Client Order ID is an optional field – one that functions as a tool for helping ActiveX API clients track orders. It is one of a number of order IDs used by the Sterling Trader® System; each of the following fields listed below is available for the purpose of tracking orders by the system:

1. ExchClOrderIS = is generated by the Sterling Trader® System; then sent to the exchange.
2. ExchOrderID = is exchange generated.
3. OrderRecordID = is an internal record ID generated by the Sterling Trader® System; it is guaranteed to be unique, relatice to other OrderRecordIDs, for multiple trading days.
4. ClOrdID is a recommended field. It is the ActiveX API client-generated Order ID used for tracking the order record. It must be designated as a unique ID, and must remain unique over multiple trading days.
5. Please remember NOT to use commas (,), equal signs (=), or tilde (˜)as part of your ClientOrder ID.

The Client Order ID must be assigned to the ClOrderID property of the STIOrder object before you call the STIOrder SubmitOrder function. This ID must be unique over multiple days. For instance, you could use the combination of Account + TimeStamp(to the millisecond) + Counter. This would provide a unique ID that would allow for orders to be sent within the same millisecond and over multiple days.

### STIEvents

*OnSTILinkSymChange* This event occurs when the symbol of a link group is changed on the front end or in the API.

*OnSTIDrop* This event is fired when a symbol is dragged out of a Sterling window and dropped.

*OnSTIOrderConfirm* This event is fired when an order submitted into Sterling Trader Pro is received by the destination.

*OnSTIOrderReject* This event is fired when an order is rejected by the Sterling Server level.

*OnSTIOrderUpdate* This event corresponds to any change on an order. The values in this update will correspond to the aggregate of the order, i.e. it will show the cumulative executed quantity rather than the single executions quantity.

*OnSTITradeUpdate* This event is fired for each execution the order received and contains the data for that execution

## Handling Rejections in the API

Orders will sometimes be rejected. In the API environment you will want to be able to handle these rejections so that they can be corrected. Within the API environment four different levels of rejection are possible. The first is the return on the SubmitOrder. Anything other than zero will be an error code. The second is the OnSTIOrderReject event. This event is triggered when the order is rejected in Sterling. The third is a backend rejection at Sterling which will change the status of the order to Rejected (STIOrderStatus = 12). This is also the case with an exchange reject, the fourth and final level of rejection.

The STIOrder SubmitOrder function will send back a return code indicating the success or failure of an order's admittance into the Sterling Trader® System. The rejected order will not appear in the trading monitor or any GUI rejections. If the return code is a negative number, then the order failed and was not sent. The negative number of the return code will correspond to one of the defined error code values that are found in the "SubmitOrder Error Codes" section on page 18. If the return code is not a negative number, then the order was sent from the Sterling Trader® System to the exchange successfully. It is absolutely necessary to include this function in your program if
it is going to be self contained and not rely on the front end for the messages.

The second level of rejections will come as the event, OnSTIOrderReject. This occurs as the order is received by the DB. However the order will not write to the trading monitor if rejected at this level. Within this event the nRejectReason will define the cause of the rejection. This will be a positive integer. This value corresponds to those listed on page 17 under STIRejectReason. The third level occurs when the value of the STIRejectReason is rrSTIAccessDenied (3). To see the cause of this you will need to view the bstrText from the OnSTIOrderReject.

The next level of rejections is the Sterling backend. This level and the fourth level, exchange rejections, operate the same. Both will be seen as part of the OnSTIOrderUpdate event. This will show as part of the nOrderStatus the STIOrderStatus of 12 is the indication that the order was rejected. To get more information on the cause of this rejection you will need to also pull The STIOrderUpdate property for bstrLogMessage. This message will be the best source for an explanation on the reject but it may not always be clear.

## Backward Compatibility and Algos

In order to make this guide as easy to understand and concise as possible some rarely used and obsolete functions were removed. These are potentially useful to those working on a special program or updating an existing project. In order to keep this data available a new backward compatibility API guide has been created.

## Sterling ActiveX Object Library

STI Order Methods
long GetQuoeCount()
long SubmitOrderStruct(structSTIOrder *pOrder)
long ReplaceorderStruct(structSTIOrder *pOrder, long nOldOrderRecId, BSTR bstroldClOrderId)

1. While both RecID and ClOrderID can be used ClOrder is the recommended option.
2. The only fields that can be changed are Price (LmtPrice, StpPrice, and PriceType) and Quantity. Other fields must match the original order, with the exception of the ClOrderID which should be unique.

HRESULT ClearOrderStruct(structSTIOrder p*Order)

| **StructSTIOrder** | | **Struct (UDT)** |
|---|---|---|
| **Properties (Read Only)** | **Type** | **Description** |
| bstrSide | BSTR | See section Side for values (page 20) |
| bstrSymbol | BSTR | The options or equities symbol. |
| bstrAccouont | BSTR | The account exactly as it appears in Sterling. |
| nPriceType | long | See Price Types for values (page 20) |
| bstrTif | BSTR | See TIF for values (page 20). |
| nQuantity | long | The number of shares (or contracts) |
| bstrDestination | BSTR | The destination as it appears on the Sterling system ex. ARCA |
| bstrClOrderIS | BSTR | Recommended field. Use this field to track orders(page 5) |
| flmtPrice | double | If price type is set to ptSTILmt this will be the limit value. |
| nDisplay | long | Send hidden (0), Reserce (qty displayed) and visible if blank |
| fDiscretion | double | Price discretion on a limit order. |
| bstrExecInst | BSTR | Filed for specifying Special Order Designations (See page 23) |
| fPegDiff | double | PEG-If price type is set to ptSSTIPegged this will be the discretion T-STP-Amount by which you want to trail the last |
| fTrailAMT | double | T-STP-The value-change needed to trigger an update. On |
| fTrailInc | double | triggerprice STP-The triggerprice on a STPor SS-STP order. |
| fStpPrice | double | MinimumFillquantity Preference |
| nMinQuantity | long | (Client-defined field) |
| bstrExecBroker | BSTR | |
| bstrUser | BSTR | O=Open, C=Closed |
| bstrCurrency | BSTR | YYYYMMDD |
| bstrOpenClose | BSTR | P=Put, C=Call |
| bstrMaturity | BSTR | Underlying equities Symbol C=Covered, |
| bstrPutCall | BSTR | U=Uncovered |
| bstrUnderlying | BSTR | For instrumental values (see page 22) |
| bstrCoverUnover | BSTR | Options strike price |
| bstrIntrument | BSTR | |
| fStrikePrice | double | Text field for Broker information. Text |
| bstrLocateBroker | BSTR | field for locate time information. |
| bstrLocateTime | BSTR | Text field for locate quantity information. The |
| nLocateQty | long | listing exchange. |
| bstrListingExchange | BSTR | RecordId of the order to be replaced. Used when splitting orders in |
| nParentRecordId | long | the order desk manager. |
| bstrBatchID | BSTR | |

**STIPosition**
Events

void OnSTIPositionUpdate (structSTIPositionUpdate* structPositionUpdate) void OnSTIPositinUpdateXML(BSTR* bstrPosition)
void OnSTIShutdown()

**Methods**
HRESULT RegisterForPositions() HRESULT DeRegisterPositions() HRESSULT GetCurrentPositions()

structSTIPositionUpdate GetPositionInfoStruct (BSTR bstrsymbol, BSTR bstrExch, BSTR btrAccount)
For bstrExch Black, '*', or 'E' = Equity 'O' = Options 'F' = Futures 'X' = Forex
long GetQueueCount()
long GetPositionList(structSTIPosUpdate()arrayPos)
long GetOptionsPosList(BSTR bstrUnderlyingSym, structSTIPosUpdate() arrayPos) long GetPosListBysym(BSTR bstrSymbol, sstructSTIPosUpdate() arrayPos)

If a symbol is not specified, GetOptionsPosList() returns all options positions; GetPosListBySym returns all positions.

**structSTIPositionUpdate**

| Properties (Read Only) | Type | Description |
|---|---|---|
| bstrSym | BSTR | Symbol |
| bstrAcct | BSTR | Trading account |
| bstrInstrument | BSTR | Position instrument |
| nOpeningPosition | long long | The position in the account to start the day. |
| nSharesBot | long long | Number of Share purchased |
| nSharesSld | long long | Number of Shares Sold. |
| nSharesSldLong | long long | (nSharesSldLong+nSharesSldShort) Number of Shares |
| nSharesSldShort | long | sold long this is a component of nSharesSld. Number of |
| nTicketsBot | double | Shares sold short this is a component of nSharesSld. Buy |
| nSticketsSld | double | side executions |
| nTicketsSldLong | double | Sell and Sell Short executions |
| nTicketsSldShort | double | Sell executions |
| fClosePrice | double | Sell Short executionos |
| fDollarsBot | double | Yesterday;s close |
| fDollarsSld | double | The total cost of BUY orders |
| fDollarsSldLong | double | The total of |
| fDollarsSldShort | long long | fDollarsSldLong+DollarsSldShort The toal |
| fPositionCost | VARIANT_BOOL | cost of SELL orders |
| fPremiumMultiplier | VARIANT_BOOL | The total cost of SHRT orders |
| fReal | | The total of fDollarsBot+fDollarsSld |
| nSharesPerContract | | This is the multipler for options positions. This is for non-integer return |
| nPremiumMultiplier | | values. This is the Realized Proft/Loss value. |
| bLast | | Options field Number of underlying shares per options contract |
| bMsgSnapShot | | This is the multiplier for options positions. For non-integers see |
| | | fPremiumMultiplier Indicates this is the last of a list of events because of a |
| | | request (GetCurrentPositions()) |
| | | Indicates that event is due to a request (GetCurrentPositions()) not a position change. |

*Not all position fields are provided some need to be calculated.*
*Example: Position = nOpeningPositiion + (nSharesBot – nSharesSld*

**STIOrderMaint**

**Methods**

HRESULT CancelOrder(BSTR bstrAccount, long OrderREcordId, BSTR bstrOldClOrderId, BSTR bstrClOrderId) HRESULT GetOrderInfo(BSTR bstrClOrderId, structSTIOrderUpdate* structorder):
HRESULT GetOrderList(VARIANT_BOOL bOpenOnly, SAFEARRAY(structSTIOrderUpdate) *arrayOrder, long *lCount) HRESULT CancelAllOrders(structSTIICancelAll*VARIANT_BOOL bExtendingOnly, BSTR bstrInstrument, BSTR bstrSymbol, BSTR bstrAccount):

A blank field in ssymbols or account specifies all symbols or accounts, bExtrendingOnly if true will only cancel orders that would extend on current positions.

HRESULT CancelSuturesOrders(BSTR bstrAccount, long OrderRecordId, BSTR bstroldClOrderId, BSTR bstrClOrderId): HRESULT CancelOptionsOrder(BSTR bstrAccount, long OrderRecordId, BSTR bstrOldClOrderId, BSTR bstrClOrderId): HRESULT CancelForexOrder(BSTR bstrAccount, long OrderRecordId, BSTR bstrOldClOrderId, BSTR bstrClOrderId): HRESULT GetEquityTradeList(SAFEARRAY(structSTITradeUpdate) *arrayTrade, long, *lCount):
HRESULT GetFuturesTradeList(SAFEARRAY(structSTITradeUpdate) *arrayTrade, long, *lCount): HRESULT GetForexTradeList(SAFEARRAY(structSTITradeUpdate) *arrayTrade, long, *lCount):
HRESULT GetFuturesOrderList(VARIANT_BOOL bOpenOnly, SAFEARRAY(structSTITradeUpdate) *arrayOrder, long,
*lCount):
HRESULT GetOptionsOrderList(VARIANT_BOOL bOpenOnly, SAFEARRAY(structSTITradeUpdate) * arrayOrder, long,
*lCount):
HRESULT GetForexOrderList(VARIANT_BOOL bOpenOnly, SAFEARRAY(structSTITradeUpdate) * arrayOrder, long,
*lCount):
HRESULT CancelOrderEx(BSTR bstrAccunt, long OrderRecordId, BSTR bstrOldClOrderId, BSTR bstrClOrderId, BSTR bstrInst, long *lRetVal);
HRESULT GetOrderListEx(structSTIOrderFilter* pFilter, SAFEARRAY(structSTIOrderUpdate) *arrayOrder, long
*lCount)
HRESULT GetTradeListEx(structSTIOrderFilter* pFilter, SAFEARRAY(structSTIOrderUpdate) *arrayOrder, long
*lCount)

1. Returns include – 16 (Pro is offline) and -37 (Multiple sub-seconod replace and/or cancel attempts)

**STIEvents**

**Events**

void OnSTILinkSymChange(structSTILink* structLink) void OnSTIDrop(structSTIDrop* structDrop) void OnSTIOrderConfirm(structSTIOrderConfirm* structOrderConfirm) void OnSTIOrderReject(structSTIOrderReject* structOrderReject)

void OnSTIOrderUpdate(structSTIOrderUpdate* structOrderUpdate) void OnSTITradeUpdate(structSTITradeUpdate* structTradeUpdate) void OnSTITradeUpdateXML(BSTR* bstrTrade)
void OnSTIOrderUpdateXML(BSTR* bstrOrder) void OnSTIOrderRejectXML(BSTR* bstrOrder) void OnSTIOrderConfirmXML(BSTR* bstrOrder) void OnSTIShutdown()

**Methods**
MeHRESULT SetOrderEventsAsStructs(bool bStruct)

**structSTIDrop** **Struct (UDT)**

| Properties (Read Only) | Type | Description |
|---|---|---|
| bstrSymbol | BSTR | Symbol you wish to link |
| bstrUnderlying | BSTR | Underlying equity symbol for Options |
| nGroup | long | Link group on Sterling you wish to link the symbol into. |

**structSTIOrderConfim** **Structs (UDT)**

| Properties (Read Only) | Type | Description |
|---|---|---|
| bstrAccount | BSTR | The name of the sterling account the order was placed |
| bstrClOrderId | BSTR | in. Trader generated order ID |
| bstrExchClOrderId | BSTR | |
| bstrExchOrderId | BSTR | |
| bstrExchOrderId2 | BSTR | |
| nstrInstrument | BSTR | The trade type (equity or option) |
| bstrMsgConfirm | BSTR | Text message for the confirm |
| nOrderRecordId | long | Order ID generated by Sterling |

**structSTIOrderReject** **Struct (UDT)**

| Properties (Read Only) | Type | Description |
|---|---|---|
| bstrAccount | BSTR | The account in which the order was sent |
| bstrBatchId | BSTR | |
| bstrClOrderId | BSTR | The user generated ID of the rejected order (C=Covered, |
| bstrCoveruncover | BSTR | U=Uncovered) |
| bstrDestinationo | BSTR | The destinatin the rejected order was sent to |
| bstrExecBroker | BSTR | |
| bstrExecInst | BSTR | |
| bstrInstrument | BSTR | |
| bstrListingExchange | BSTR | |
| bstrMaturity | BSTR | (YYYYMMDD) |
| bstrpenClose | BSTR | (O=Open, C=Close) |
| bstrPriceType | BSTR | The price type from the rejected order (P=Put, C=Call) |
| bstrPutCall | BSTR | The side of the rejected order The symbol in |
| bstrSide | BSTR | the rejected order |
| bstrSymbl | BSTR | The time in force of the rejected order Contains text |
| bstrTif | BSTR | information on the Rejection |
| bstrText | BSTR | The underlying symbol of the rejected options order (Client- |
| bstrUnderlyin | BSTR | Defined field) |
| g bstrUser | BSTR | |
| fDiscretion | double | The limit price of the rejected order |
| fLmtPrice | double | |
| fPegDiff | double | |
| fStpPrice | double | |
| fStrikePrice | double | |
| fTrailAmt | double | |
| fTrailInc | double | The display quantity of the rejected order |
| nDisplay | long | |

## StructSTIOrderUpdate Structs (UDT)

| Properties (Read Only) | Type | Description |
|---|---|---|
| bstrAccount | BSTR | |
| bstrAction | BSTR | |
| bstrBatchId | BSTR | |
| bstrClOrderId | BSTR | |
| bstrCoverUncover | BSTR | (C=Covered, U=Uncovered) |
| bstrDestination | BSTR | |
| bstrExchClOrderId | BSTR | |
| bstrExchClOrderId | BSTR | |
| 2 bstrExchOrderid | BSTR | |
| bstrExecBroker | BSTR | |
| bstrInstrument | BSTR | |
| bstrLogMessage | BSTR | Provides the log messages on an order (YYYYMMDD) |
| bstrMaturity | BSTR | (O=Open, C=Close) (P=Put, C=Call) |
| bstrOpenClose | BSTR | |
| bstrOrderTime | BSTR | |
| bstrPriceType | BSTR | |
| bstrPutCall | BSTR | |
| bstrSide | BSTR | (Clinet-defined field) (Trader/Login ID) |
| bstrSymbol | BSTR | |
| bstrTif | BSTR | |
| bstrUnderlying | BSTR | |
| bstrUpdateTime | BSTR | |
| bstrUser | BSTR | |
| bstrUserId | BSTR | |
| fAvgExecPrice | double | |
| fDiscreption | double | |
| fLmtPrice | double | |
| fPegDiff | double | |
| fStpPrice | double | |
| fStrikePrice | double | |
| fTrailAmt | double | |
| fTrailInc | double | |
| fUrStpdPrice | double long | |
| nCumExecQuantity | long long | |
| nDbsNo | long long | |
| nDisplay | long long | |
| nLvsQuantity | long | |
| nMinQuantity | long | |
| nOrderRecordId | long | |
| nOrderStatus | long | |
| nPriceTyp | VARIANT_BO | |
| e | OL | |
| nQuantity | | |
| nSeqNo | | |
| nTrailId | | |
| bSvrrStpR | | |
| eleased | | |

## StructSTITradeUpdate

| Properties (Read Only) | Type | Description |
|---|---|---|
| bstrAccount | BSTR | |
| bstrAction | BSTR | 'A'=Add, 'C'=Change, 'D'=Delete |
| bstrBatchId | BSTR | Used by Order Dest to associate a group of orders Contra |
| bstrClOrderId | BSTR | |
| bstrContra | BSTR | Banker |
| bstrCoverUncover | BSTR | |
| bstrDestination | BSTR | |
| bstrExchClorderId | BSTR | |
| bstrExchExecId | BSTR | |
| bstrExchOrderId | BSTR | |
| bstrExchOrderId2 | BSTR | |
| bstrExecBroker | BSTR | |
| bstrExecInst | BSTR | |
| bstrInstrument | BSTR | |
| bstrLiquidity | BSTR | |
| bstrLogMessage | BSTR | |
| bstrMaturity | BSTR | |
| bstrOpenClose | BSTR | |
| bstrOrderTime | BSTR | |
| bstrPutCall | BSTR | |
| bstrSide | BSTR | |
| bstrSpecialist | BSTR | |
| bstrSymbol bstrTif | BSTR | |
| bstrTradeTime | BSTR | |
| bstrUnderlying | BSTR | |
| bstrUpdateTime | BSTR | |
| bstrUserId | BSTR | |
| fDiscretion | BSTR | |
| fExecPrice | double | |
| fLmtPrice | double | |
| fPegDif fStpPrice | double | |
| fStrikePrice | double | |
| nDbsNo | double | |
| nLvsQuantity | double | Database number |
| norderRecordId | long long | |
| nPriceType | long long | |
| nQuantity | long long | |
| nmSeqNo | long | |
| nTradeRecordId | VARIANT_BOOL | |
| nClearable | VARIANT_BOOL | |
| nEcnFee | | |

## STIApp

**Methods**
HRESULT SwitchLinkGroupSymbol(long nLinkGroup, BSTR bstrSym, BSTR brstExch)

- Sends the symbol into Sterling
Long GetDestinationList(BSTR() arrayDests)
- Oull a list of available destinations BSTR GetTraderName()
1 See STIAcctMaint (on page 17) for GetAccountList()
- Provides the login name of the user, often the same as the account/ BSTR GetServerTime()
Pulls the Time off Sterling DB in this format: CCYYMMDDhhmms

SetModeXML(bool bXML)
Enables the use of XML events.
VARIANT_BOOL ISApiEnabled()
Confirm with API that the trader is entitled to use API.

**STIAcctMaint**
**Events**
void OnSTIAcctUpdate(structSTIAcctUpdate* structAcctUpdate) void
OnSTIAcctUpdateXML(BSTR*bstrAcct)
void OnSTIShutdown()

**Methods**
long GetQueueCount()
long GetAccountList(BSTR() arrayAccts)
STIAcctHRESULT ClearAccountUpdateStruct(structSTIAcctUpdate*pAcctUpdate) HRESULT
Destroy()

**structSTIOrderFilter** **Struct (UDT)**

| Properties | Type | Description |
|---|---|---|
| bstrInstrument bstrSymbol | BSTR BSTR | |
| bstrAccount | BSTR | |
| bOpenOnly | VARIANT_BOOL | |

**structSTITradeFilter** **Struct (UDT)**

| Properties | Type | Description |
|---|---|---|
| bstrInstrument bstrSymbol | BSTR BSTR | |
| bstrAccount | BSTR | |

**STIRejectReason** **Enums**

| Value | Reject | Description |
|---|---|---|
| 0 | rrSTIUnknown | |
| 1 | rrSTIUnknownPid | |
| 2 | rrSTIInvalidPassword | |

| | |
|---|---|
| 3 | rrSTIAccessDenied |
| 4 | rrSTINotFound |
| 5 | rrSTICannotRoute |
| 6 | rrSTIPendingCancel |
| 7 | rrSTIPendingReplace |
| 8 | rrSTIOrderClosed |
| 9 | rrSTICannotCreate |
| 10 | rrSTIDupeClOrdId |
| 11 | rrSTINoSeqNoAvailable |
| 12 | rrSTIInvalidAcct |
| 13 | rrSTIInvalidDest Sending a destination that the trader is not enabled for will trigger this. |
| 14 | rrSTIError |
| 15 | rrSTIDupeSeqNo |
| 16 | rrSTINoChange |
| 17 | rrSTIInvalidSeqNo |
| 18 | rrSTIInvalidQty |
| 19 | rrSTITltc Too late to cancel |
| 20 | rrSTIShareLimit |
| 21 | rrSTIDollarLimit |
| 22 | rrSTIBuyingPower |
| 23 | rrSTITenSecRule |
| 24 | rrSTINotSupported |
| 25 | rrSTIDupeAcct |
| 26 | rrSTIInvalidGroupId |
| 27 | rrSTIDupeStation |
| 28 | rrSTIPosTradingLmt |
| 29 | rrSTITltcMoc Too late to cancel MOC |
| 30 | rrSTIHardToBorrow |
| 31 | rrSTIVersion |
| 32 | rrSTIDupeLogin |
| 33 | rrSTIInvalidSym |
| 34 | rrSTINxRules |
| 35 | rrSTIBulletNotRequired |

| | |
|---|---|
| 36 | rrSTIMocMktImb |
| 37 | rrSTINx30SecRule |
| 38 | rrSTIEasyToBorrowOnly |
| 39 | rrSTIStaleOrder |
| 40 | rrSTILast |

## SubmitOrder Error Codes Values

| Value | Error | Description |
|---|---|---|
| 0 | No Errors | Order has been accepted by the GUI |
| -1 | Invalid Account | The account used is not permissioned for the login Not a valid side see page 20 |
| -2 | Invalid Side | |
| -3 | Invalid Qty | |
| -4 | Invald Symbol | |
| -5 | Invalid Price Type | |
| -6 | Invalid Tif | |
| -7 | Invalid Destination | Not including a destination in the order will trigger this |
| -8 | Exposure Limit Violation | |
| -9 | NYSE+ Rules Violation | |
| -10 | NYSE+ 30-Second Violation | |
| -11 | Diasble SelectNet Short Sales | |
| -12 | Long Sale Position Rules Violation | |
| -13 | Short Sale Position Rules Violation | Orders will not be split but a side change will occur |
| -14 | GTC Orders Not Enabled | Orders will not be split but a side change will occur |
| -15 | ActiveX API Not Enabled | |
| -16 | Sterling Trader□ Pro is Offline | |
| -17 | Security Not Marked as Located | |
| -18 | Order Size Violation | |
| -19 | Position Limit Violation | |
| -20 | Buying Porwer/Margin Control Violationo P/L Control Violation | |
| -21 | Account Not Enabled for this Product | |
| -22 | Trader Not Enabled for Futures Minimum | |
| -23 | Balance Violatioin | |
| -24 | Trader not Enabled for odd lots | |
| -25 | Order dollar limit exceeded | Open or cover transactions |
| -26 | Traded Not Enabled for Options | |
| -27 | Soft Share limit exceeded | |
| -28 | Loss from max profit control violation (Title builds only) | |
| -29 | Desk Quantity enforcement violation | |
| -30 | Account not enabled for Sell to Open (Options) | |
| -31 | Account allowed to. 'Close/Cxl' only | |
| -32 | Trader not enabled for security locating | |
| -33 | Order not able to obe replaced (ReplaceOrder only) | |
| -34 | Trader not enabled for security locating | |
| -35 | Invalid Maturity date | |
| -36 | Only one cancel and/or replace allowed per order per second | |
| -37 | Account's maximum position value for this symbol exceeded | |
| -38 | Symbol violates the account's mini/max price settings | |
| -39* | Quote unavailable to calculate order dollar limit | |
| -40* | Quote unavailable to calculate maximum position cost | |
| -41* | Quote unavailable to calculate buying power | |
| -42* | Quote unavailable to calculate margin control | |
| -43* | Floatiing BP violatiions | |
| -44 | Market order would remove liquidity (front end settings) | |
| -45 | Not enabled for server stop orders | |
| -46 | Not enabled for trail stop orders | |
| -47 | Order would exceed the max open orders per | |
| -48 | | |
| -49* | | |

| | | |
|---|---|---|
| | side on this symbol | |
| -50 | Quote unavailable or compliance threshold exceeded or quote unavailable | |
| -51 | Neither last nor close price available for MKT order | |
| -52 | Quote unavailable or does not meet min average daily volume | |
| -53 | CLO Orders not allowed | |
| -54 | Option Position Effect Error | |
| -55 | Funari | |
| -56 | Invalid Desk Price Invalid Price Fields | |
| -57 | Board Lot | |
| -58 | No Quote Board Lot | |
| -59 | Price Tick | |
| -60 | No Tick Price Tick | |
| -61 | No Quote Price Tick | |
| -62 | On Open/OPG Disabled | |
| -63 | Order Change Display | |
| -64 | No Quote | |
| -65 | Stop Orders | |
| -66 | Option Risk Levels | |
| -67 | Account Disabled | |
| -68 | Cannot Trade Options on BP Control | |

**Note: A quote is needed to calculate the values for symbol min/max price setting, dollar limit, max position cost, buying power, margin control and the compliance threshold. In order to prevent this rejection simply register for the quotes using the composite (*) on the symbol before you send the order*

## STIOrderStatus Enums

| Value | Order Status | Description |
|---|---|---|
| 0 | osSTIUnknown | |
| 1 | osSTIPendingCancel | Cancel request received by Sterling DB byt the UROUT is pending |
| 2 | osSTIPendingReplace | Replace received by Sterling DB but the replaced UROUT is pending |
| 3 | osSTIDoneForDay | |
| 4 | osSTICalculated | |
| 5 | osSTIFilled | Order is completely filled |
| 6 | osSTStopped | |
| 7 | osSTISuspended | |
| 8 | osSTICanceled | Order has been confirmed as cancelled |
| 9 | osSTIExpired | |
| 10 | osSTIIPartiallyFilled | Fill. Has been received byt the order is not completed |
| 11 | osSTIReplaced | Order has been confirmed as replaced |
| 12 | osSTIRejected | Order is rejected |
| 13 | osSTINew | Order received and conofirmed (open order) |
| 14 | osSTIPendingNew | Order received on DB but not confirmed |
| 15 | osSTIAcceptedForBidding | |
| 16 | osSTIAdjusted | Order has been manually updated |
| 17 | osSTIStatused | |

**STIPriceTypes** **Enums**

| Value | Price Type | Description |
|---|---|---|
| 1 | ptSTIMkt | Market order |
| 2 | ptSTIMktClo | Market on close order |
| 3 | ptSTIMktOb | Market or better |
| 4 | ptSTIMktWow | Market withut waiting |
| 5 | ptSTIlmt | Limit |
| 6 | ptSTILmtClo | Limit on close |
| 7 | ptSTILmtStp | Stop order |
| 8 | ptSTILmtStpLmt | Stop limit order |
| 9 | ptSTILmtOb | Limit or better |
| 10 | ptSTIWow | Without waiting |
| 11 | ptSTILmtWow | Limit without waiting |
| 12 | ptSTIBas | NYSE basis order |
| 13 | ptSTIClo | Close order |
| 14 | ptSTIPegged | Peg order |
| 100 | ptSTISvrStp | Server side stopo order |
| 101 | ptSTISvrStpLmt | Server side stop limit order |
| 102 | ptSTITrailStp | Trailing stop order |

**Side Values**

| Value | Meaning |
|---|---|
| 'B' | =BUY |
| 'C' | =BUY TO COVER |
| 'S' | =SELL |
| 'T' | =SSHRT |
| 'M' | =BUY- |
| 'P' | =SELL+ |
| 'E' | =SSHRTEX |

**Tifs Values**

| Value | Maning |
|---|---|
| 'D' | =DAY |
| 'G' | =GTC |
| 'X' | =GTX |
| 'F' | =FOK |
| 'I' | =IOC |
| 'O' | =OPG |
| 'E' | =EXT |
| '1' | =OS |
| 'A' | =AEX (Auto-x) |
| 'N' | =NOW |

**Action**

| Value | Meaning | Description |
|---|---|---|
| 'A' | = Add | |
| 'C' | = Change | |
| 'D' | = Delete | |
| 'S' | = Status | |

**MaintainAccount Error Codes Values**

| Value | Invalid Field |
|---|---|
| 0 | No errors |
| -1 | Pro is offline |
| -2 | Traders are not allowed to maintain accounts |
| -3 | Invalid account |
| -4 | Manager is not entitled to change fields |

**Instrument Values**

| Value | Meaning |
|---|---|
| "B" | Bullet Order Bullet Trade Equity Order Equity |
| "Non-B" | Trade Equity |
| "E" | Options Futures |
| "O" | Forex |
| "F" | |
| "X" | |

**NxRules (NYSE+ Rules Enforcement) Values**

| Value | Meaning |
|---|---|
| "0" | Use Sterling. Trader Pro default settings (NYSE+ |
| "1" | rules violation settings) Convert the destination |
| "2" | to NYSE is there is a NYSE+ rules violation Reject the order if there is a NYSE+ rules violation |

**Special Order Designations Values**

| ExecInst | Meaning |
|---|---|
| 'E' | DNI DNR AON |
| 'F' | Pegged Mid-Market Pegged Market Pegged |
| 'G' | Primary |
| 'M' | Pegged Best |
| 'P' | |
| 'R' | |
| 'T' | |

**ExecInst (ARCA only) Meaning**

| | |
|---|---|
| '2' | Sweep Reserve |
| '6' | Post no Preference |

**Note 1:** If you want more than one at a time, use thgem together separated by a single space between each. Order
does not matter. Example: ExecInst = 'E F G' (for DNI, DNR, AON)