DeltaExchange

STage GOM 사용도움말

DeltaExchange

1. GOM 의 소개

GOM 이란?

고수 Plus 에서는 COM(Component Object Model), OLE Automation 기술을 이용하여 고수 내부의 객체들을 공개하고 있습니다. 이 공개된 객체정보가 Gosu Object Model(GOM)입니다. GOM 에서 제공되는 정보를 이용하여 사용자가 원하는 전략을 개발할 수 있습니다.

2. STage 에서 GOM 의 사용

STage 와 GOM 의 만남

이제 STage 에서 GOM 을 사용할 수 있게 되었습니다. STage 상에서 고수 내부 객체에 접근이 가능하며, 계좌정보, 주문정보, 종목정보 등을 가져올 수 있습니다.

STage 에서는 고수 Plus 의 실시간 자료를 STage 에서 모두 활용할 수 있습니다. 또한 기존에 GOM 을 이용하기 위해서는 COM 기술과 프로그래밍에 대한 숙지가 필요하였으나, STage 사용만으로 GOM 의 모든 것을 이용할 수 있습니다. 고수 Plus 의 내부데이타를 활용하여, 시장상황의 변화를 바로 적용할 수 있는 전략 구축이 가능합니다. GOM 을 통해 그 동안 불가능으로만 여겼던 다양한 전략들을 수행할 수 있을 것입니다.

GOM을 이용해 구현할 수 있는 기능들

계좌정보의 실시간 불러오기가 가능합니다. 예탁금 총액 및 증거금 총액, 주문 가능 총액 등의 조회를 실시간으로 할 수 있으며 당일 실현/평가 손익의 실시간 조회가 가능합니다. 이를 활용하여 계좌금액기준 및 당일 손실액 기준 손절기능을 비롯하여 사용자가 원하는 다양한 기능을 구현할 수 있습니다.

국내 선물 및 옵션에 대한 시세조회가 가능합니다. 현재가, 틱체결량, 매수/매도호가 잔량 및 건수 등을 실시간으로 불러올 수 있습니다. 또한 [STage 차트], [STage 시스템 주문]과 별도로 체결주문을 수행할 수도 있습니다.

3. GOM 시작하기

STage 에서 GOM 을 사용하는 방법을 다음의 순서에 따라 살펴보겠습니다.

1) STage 에 GOM 을 연결하기
2) GOM 구조 이해하기
3) 메소드를 이용하여 GOM 객체 접근하기

(1) STage 에 GOM 을 연결하기

GOM 을 STage 에서 사용하기 위해서는 지표를 작성할 때 아래의 코드를 추가하여야 합니다.

```
Object(NoReAlloc): oGom(DLL("STage_Root", "GosuGOM.dll", ""))
```

위 문장을 지표작성 시 추가하게 되면, GOM 을 STage 에서 사용할 수 있게 됩니다. GOM 객체에 접근할 때 사용하게 되는 이름은 **oGOM** 이 됩니다. Object(NoReAlloc)과 DLL("STage_Root", "GosuGOM.dll", ""))에 관한 내용은 문서 뒷부분의 상세내용에서 다루겠습니다.

(2) GOM 구조 이해하기

고수 홈페이지(http://thegosu.com) 의 [GOM] – [GOM 구조도]를 보시면 GOM 의 객체와 구조가 상세히 안내되어 있습니다. (http://thegosu.com/template/2_00004_popup2.html) [그림 1]은 GOM 의 구조도 입니다. GOM 의 구조는 트리구조를 이루고 있습니다. 즉, 사용자가 특정 객체 및 정보에 접근하기 위해서는 상위객체에서 하위객체로 접근해야 한다는 것입니다. GOM 구조에 대한 자세한 내용은 다음 장 **4. GOM 구조, 메소드, 이벤트** 에서 이어집니다.

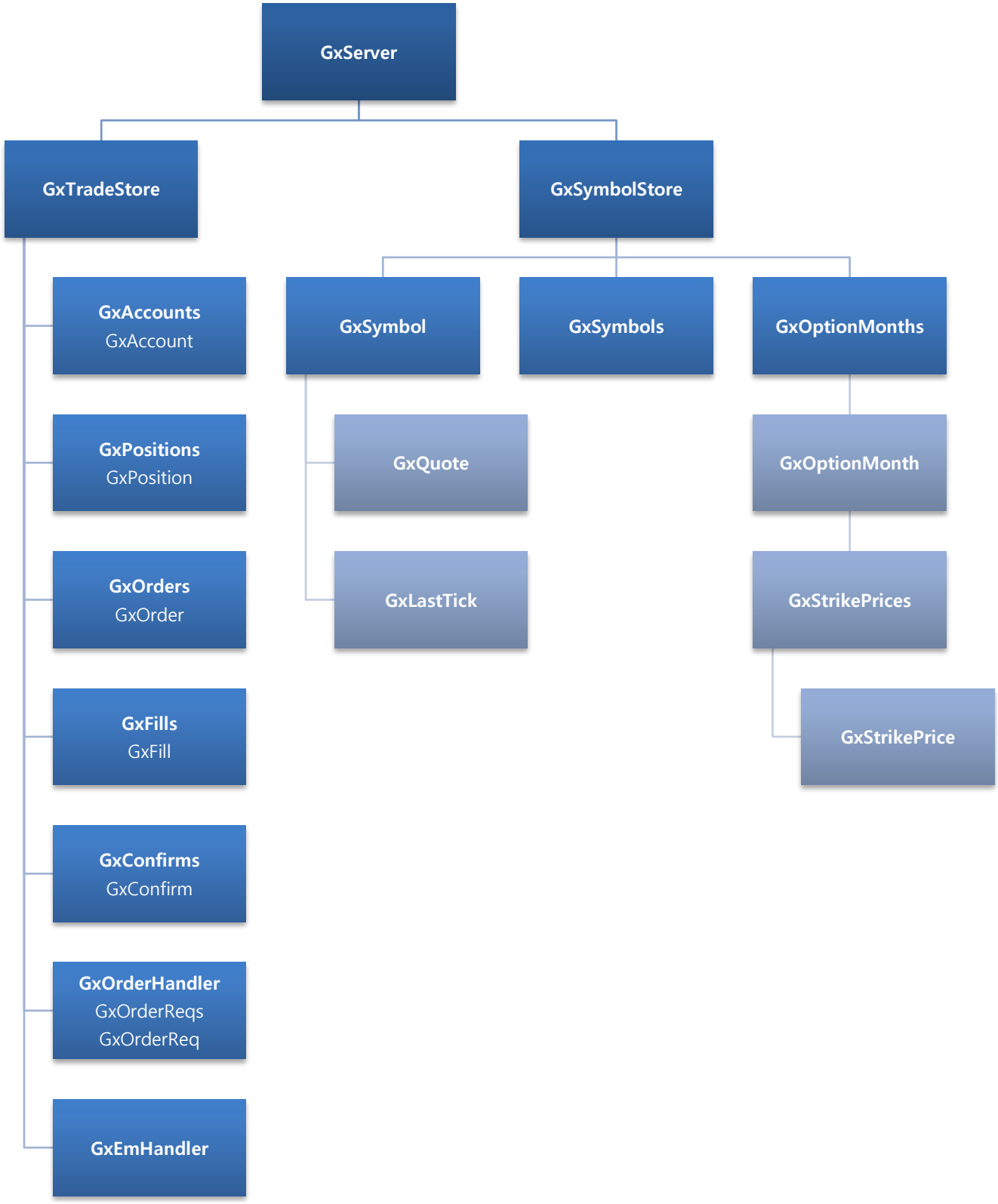
GOM 의 최상위 객체는 GxServer 이며, 바로 아래 하위객체로 GxTradeStore 와 GxSymbolStore 가 있습니다. GxTradeStore 는 계좌, 포지션, 주문실행, 주문확인 등의 객체와 정보를 담고 있으며 GxSymbolStore 는 선물, 옵션 각각의 종목 시세 객체와 정보를 담고 있습니다.

예를 들면 개별 계좌정보에 관련한 객체는 GxAccount 인데, 이 객체에 접근하기 위해서는 아래의 순서에 따라 접근해야 합니다.

GxServer -> GxTradeStore –> GxAccounts -> GxAccount

STage 에서는 GOM 의 GxChartStore 를 제외한 GxTradeStore, GxSymbolStore 의 하위 객체에 접근이 가능합니다.

[그림 1 : GOM 구조도]


GxServer
GxTradeStore
GxSymbolStore
GxAccounts
GxAccount
GxSymbol
GxSymbols
GxOptionMonths
GxPositions
GxPosition
GxQuote
GxOptionMonth
GxOrders
GxOrder
GxLastTick
GxStrikePrices
GxFills
GxFill
GxStrikePrice
GxConfirms
GxConfirm
GxOrderHandler
GxOrderReqs
GxOrderReq
GxEmHandler

(3) 메소드를 이용하여 GOM 객체 접근하기

(1) STage 에 GOM 을 연결하기를 통해 oGom 을 생성하고 나면 다음과 같은 메소드를 사용할 수 있습니다. 아래의 메소드를 사용하여 각각의 GOM 객체에 접근할 수 있습니다. 메소드에 대한 상세한 내용은 다음 장 **4. GOM 구조, 메소드, 이벤트** 에서 이어집니다.

메소드	Input(Object)	Input(Field ID)	설명
GetObject	Num	Num	객체를 조회합니다.
GetObjectByIndex	Num	Num	순서에 따라 객체를 조회합니다.
GetObjectVal	Num	Num	객체에 접근하여 숫자정보를 조회합니다.
GetObjectStr	Num	Num	객체에 접근하여 문자열정보를 조회합니다.
GetObjectBool	Num	Num	객체에 접근하여 Boolean 변수를 조회합니다.

GxTradeStore = oGom.GetObject(GxServer, 3)

앞서 생성한 oGom 오브젝트에서 GetObject 메소드를 호출합니다. GetObject 메소드는 두 개의 값을 입력 받습니다. 첫 번째 입력은 "대상객체"이며 두 번째 입력은 "필드 ID"입니다.

GxServer	
Field ID	내용
3	GxTradeStore
4	GxSymbolStore

GxServer 의 필드는 위와 같습니다. oGom.GetObject(GxServer, 3)에서 GxServer 가 대상객체이며 GxServer 객체의 필드 3 번에 해당하는 객체를 조회합니다. 위의 구문을 통해 GxTradeStore 는 GOM 구조도 상의 GxTradeStore 객체를 가르키는 변수가 됩니다.

```
GxAccounts = oGom.GetObject(GxTradeStore, 1)
Count = oGom.GetObjectVal(GxAccounts, 3)
```

마찬가지로 첫 번째 문장은 GxTradeStore 객체의 1 번 필드에 접근합니다. 그렇게 해서 GxAccounts 객체에 접근할 수 있게 되며, 두 번째 문장은 GxAccounts 의 3 번 필드의 값을 조회하여 Count 에 저장합니다. GxAccounts 의 3 번 필드는 Numeric 으로 "총 계좌수" 정보를 담고 있습니다. 이와 같이 메소드를 통해 객체에 단계별로 접근하여 원하는 정보를 조회할 수 있습니다.

GOM 객체 정보를 차트에 표시하기

텍스트 오브젝트를 이용하면 GOM 객체 정보를 차트에 나타낼 수 있습니다. 아래와 같이 oTxt1 이라는 텍스트 오브젝트를 선언 및 생성하여 DailyPL 이라는 Numeric 변수를 STage 차트에 표시할 수 있습니다.

```
Object(NoReAlloc, NoRecover): oTxt1(Text(D,T,H,""))

oTxt1.Text = "당일 총손익  " & NumToStr( DailyPL, 0 )
oTxt1.SetLocation(d,t,h)
```

이외에도 Print()함수를 이용하여 STage 전략편집기의 출력 창에서 결과값을 확인할 수 있습니다. 텍스트 오브젝트의 자세한 사용법은 STage 도움말을 참고하시기 바랍니다.

GOM 사용 지표 작성 시 주의할 점

GOM 사용은 실시간 트레이딩을 의미합니다. 그렇기 때문에 지표 코드가 반복실행 등으로 예기치않은 결과를 초래하지 않도록 지표 작성 시 각별한 주의를 하셔야 합니다. 반복실행을 방지 할 수 있는 방법으로는 Boolean 변수를 사용하는 방법이 있습니다.

```
Vars(NoRecover, NoReInit): xbFirst(True)

If xbFirst Then

        // 코드 내용

        xbFirst = False

End If
```

위와 같이 Boolean 변수와 If 문을 사용하면 If 문 안의 코드가 1 회만 수행되도록 할 수 있습니다. 지표 작성 시, 이러한 방법을 반드시 사용하셔야 합니다.

4. GOM 구조, 메소드, 이벤트

이번 장에서는 GOM 구조, 메소드, 이벤트를 자세히 살펴보겠습니다.

(1) GOM 구조 – 객체, 데이터, 이벤트

GxServer

GOM 의 최상위 클래스입니다.

ID	변수유형	사용가능 메소드	필드명	설명
1	String	GetObjectStr	LastMessage	최신 에러 정보
2	Numeric	GetObject	ServerInfo	서버 연결 정보를 조회하고, 서버와 연결이 끊어졌을 경우 이를 알려주는 역할을 합니다.
3	Numeric	GetObject	TradeStore	주문, 계좌조회, 포지션 조회 등 거래 전반 데이터 객체와 주문 도구 객체를 제공합니다.
4	Numeric	GetObject	SymbolStore	종목을 종합적으로 관리, 종목을 참좋라 수 있는 다양한 방법을 제공합니다.

GxTradeStore

주문, 계좌조회, 포지션조회 등 거래 전반 데이터 오브젝트들의 인터페이스와 주문 도구 오브젝트들의 인터페이스를 제공합니다.

ID	변수유형	사용가능 메소드	필드명	설명
1	Numeric	GetObject	Accounts	계좌목록을 관리하는 Collection 객체를 반환합니다.
2	Numeric	GetObject	Positions	포지션목록을 관리하는 Collection 객체를 반환합니다.
3	Numeric	GetObject	Orders	주문목록을 관리하는 Collection 객체를 반환합니다.
4	Numeric	GetObject	Fills	체결목록을 관리하는 Collection 객체를 반환합니다.
5	Numeric	GetObject	Confirms	정정/취소 확인목록을 관리하는 Collection 객체를 반환합니다.
6	Numeric	GetObject	OrderHandler	주문을 전송하는데 사용하는 객체입니다.
7	Numeric	GetObject	EmHandler	서버 장애로 인하여 주문접수 및 체결 결과가 원할하게 전달이 되지 않을 때 긴급 주문 및 강제로 서버로 조회를 수행하는 객체입니다.
8	String	GetObjectStr	LastMessage	최신 에러 정보

GxAccounts

계좌 목록을 관리하고, 계좌 관련된 이벤트를 발생시킵니다.

ID	변수유형	사용가능 메소드	필드명	설명
3	Numeric	GetObjectVal	Count	총 계좌 수
0	Numeric	GetObjectByIndex GetObjectByStr	GxAccount – By Index, Str	개별 계좌를 순서 또는 코드로 조회합니다. GetObjectByIndex 메소드를 사용해야 합니다.
4	String	GetObjectStr	LastMessage	최신 에러 내용
1	Event	RegEvent	OnAccountUpdated	계좌의 속성값(예탁금, 증거금, 손익, 수수료 등)이 변경될 때 발생합니다.

GxAccount

계좌 정보를 담고 있는 객체입니다.

ID	변수유형	사용가능 메소드	필드명	설명
1	String	GetObjectStr	Code	'-'이 제외된 계좌번호
2	String	GetObjectStr	Desc	계좌명
3	String	GetObjectStr	HiphenedCode	'-'이 들어간 형태의 계좌번호
4	Bool	GetObjectBool	Authenticated	비밀번호 확인 여부
5	Bool	GetObjectBool	ApplyBiasMargin	해당 계좌가 편향증거금 대상인가를 조회
6	Numeric	GetObjectVal	DepositTotal	예탁금 총액
7	Numeric	GetObjectVal	DepositCash	예탁금 현금
8	Numeric	GetObjectVal	MarginTotal	증거금 총액
9	Numeric	GetObjectVal	MarginCash	증거금 현금
10	Numeric	GetObjectVal	LiquidTotal	주문 가능 총액
11	Numeric	GetObjectVal	LiquidCash	주문 가능 현금
12	Numeric	GetObjectVal	Commission	잠정 수수료
13	Numeric	GetObjectVal	DailyPL	당일 총 손익 : 실현손익(AccPL) + 평가손익(EvalPL)
14	Numeric	GetObjectVal	AccPL	당일 누적 실현(청산) 손익
15	Numeric	GetObjectVal	EvalPL	미결제 약정의 평가 손익
16	Numeric	GetObject	Orders	해당 계좌의 주문 목록 객체
17	Numeric	GetObject	Positions	해당 계좌의 포지션 목록 객체
18	String	GetObjectStr	LastMessage	최신 에러 내용

GxPositions

포지션 목록을 관리하며, 포지션의 변동시 통보하는 기능을 합니다.

ID	변수유형	필드명	설명
3	Numeric	Count	포지션 수
0	Numeric	GxPosition – By Index	개별 포지션을 순서에 따라 조회합니다. GetObjectByIndex 메소드를 사용해야 합니다.
4	String	LastMessage	최신 에러 메시지
1	Event	RegEvent	새로운 포지션이 추가될 때 발생합니다.
2	Event	RegEvent	포지션의 수량이 변경되면 발생합니다.
3	Event	RegEvent	포지션의 평가 손익이 변경되었을 때(포지션 종목의 현재가 변경) 발생합니다.

GxPosition

개별 포지션에 대한 모든 정보를 담고 있는 객체입니다.

ID	변수유형	사용가능 메소드	필드명	설명
1	Numeric	GetObject	Account	포지션의 계좌 객체
2	Numeric	GetObject	Symbol	포지션의 종목 객체
3	Numeric	GetObjectVal	Qty	최종 포지션 수량 (매수 : 양수, 매도 : 음수)
4	Numeric	GetObjectVal	PrevQty	전일 최종 포지션 수량
5	Numeric	GetObjectVal	TodayQty	당일(변동) 포지션 수량
6	Numeric	GetObjectVal	AvgPrice	평균 단가
7	Numeric	GetObjectVal	EvalPL	평균 손익
8	Numeric	GetObjectVal	ShortOrderQty	매도 미체결 주문 수량
9	Numeric	GetObjectVal	LongOrderQty	전일 최종 포지션 수량
10	String	GetObjectStr	LastMessage	최신 에러 정보

GxOrders

주문 목록을 관리하며, 주문 관련 변경시 통보하는 역할을 합니다.

ID	변수유형	사용가능 메소드	필드명	설명
3	Numeric	GetObjectVal	Count	주문 목록 수
0	Numeric	GetObjectByIndex	GxOrder – By Index	순서에 따라 주문 객체를 조회합니다. GetObjectByIndex 메소드를 사용해야 합니다.
4	String	GetObjectStr	LastMessage	최신 에러 메시지

1	Event	RegEvent	OnNewOrder	새로운 주문이 추가되면(증전접수) 발생합니다.
2	Event	RegEvent	OnOrderUpdated	체결, 정정, 취소 등으로 주문의 상태가 변경되면 발생합니다.

GxOrder

개별 주문에 대한 정보 및 상태를 담고 있는 객체입니다. 주문접수부터 정정, 취소, 체결에 대한 정보를 모두 이 객체를 통해 확인할 수 있습니다. 주문이 거래소에 접수되었을 때 생성됩니다.

ID	변수유형	사용가능 메소드	필드명	설명
1	Numeric	GetObject	Account	주문 대상 계좌 객체
2	Numeric	GetObject	Symbol	주문 대상 종목 객체
3	Numeric	GetObject	Position	주문에 대한 포지션 객체
4			OrderType	주문 종류(신규/정정/취소)
5	Numeric	GetObjectVal	PositionType	매수/매도 구분 (0 : 매수 타입 / 1 : 매도 타입)
6	Numeric	GetObjectVal	PriceType	주문가격유형(지정가/시장가/조건부지정가/최유리지정가) (0 : 지정가 / 1 : 시장가 / 2 : 조건부지정가 / 3 : 최유리지정가)
7	Numeric	GetObjectVal	Qty	주문 수량(정정 및 취소에 의해 감소됨)
8	Numeric	GetObjectVal	InitQty	최초 주문수량(정정 및 취소에 의해서 변하지 않음)
9	Numeric	GetObjectVal	Price	주문 가격
10	Numeric	GetObjectVal	OrderState	주문 상태
11	Numeric	GetObjectVal	SrvAcptNo	서버 접수 번호
12	Numeric	GetObjectVal	KseAcptNo	거래소 접수번호
13	Numeric	GetObjectVal	KseAcptTime	거래소 주문 접수시간
14	Numeric	GetObjectVal	FillQty	체결 수량
15	Numeric	GetObjectVal	LastFillPrice	가장 최근 체결가
16	Numeric	GetObjectVal	UnFillQty	미체결 수량
17	Numeric	GetObject	TargetOrder	정정 및 취소의 대상 주문 객체
18	Numeric	GetObjectVal	TargetSrvAcptNo	정정 및 취소 대상 주문의 서버 접수번호
19	Numeric	GetObjectVal	TargetKseAcptNo	정정 및 취소 대상 주문의 거래소 접수번호
20	Numeric	GetObject	Fills	본 주문에 대해 발생한 체결 목록 객체
21	Numeric	GetObject	Confirms	본 주문이 정정 및 취소 주문일 경우 발생한 확인목록 객체
22	String	GetObjectStr	LastMessage	최신 에러 정보

GxFills

체결 목록을 관리하고, 체결이 발생하거나 체결 내용이 변경되면 이를 알립니다.

ID	변수유형	사용가능 메소드	필드명	설명
3	Numeric	GetObjectVal	Count	포지션 수
0	Numeric	GetObjectByIndex	GxFill – By Index	목록에서 체결 객체를 순서에 따라 참조합니다. GetObjectByIndex 메소드를 사용해야 합니다.
4	String	GetObjectStr	LastMessage	최신 에러 메시지
1	Event	RegEvent	OnNewFill	신규 체결시 통보합니다.
2	Event	RegEvent	OnFillUpdated	체결정보 수정시 통보합니다.

GxFill

체결에 대한 정보를 담고 있는 객체입니다. 체결이 될 때마다 생성이 됩니다. 단, 동일 주문에 대한 체결이 10 초 이내에 두 개 이상 들어올 때는 한 개의 체결로 처리합니다.

ID	변수유형	사용가능 메소드	필드명	설명
1	Numeric	GetObject	Order	체결 대상 주문 객체
2	Numeric	GetObjectVal	FillNo	거래소 체결번호
3	Numeric	GetObjectVal	FillQty	체결수량
4	Numeric	GetObjectVal	FillPrice	체결가
5	Numeric	GetObjectVal	FillTime	체결시간
6	Numeric	GetObjectVal	NearPrice	근월물 체결가
7	Numeric	GetObjectVal	FarPrice	원월물 체결가
8	String	GetObjectStr	LastMessage	최신 에러 내용

GxConfirms

정정 확인 및 취소 확인을 관리하며, 새로운 확인 발생시 이를 통보합니다.

ID	변수유형	사용가능 메소드	필드명	설명
1	Numeric	GetObjectVal	Count	확인 객체 수
2	Numeric	GetObjectByIndex	GxConfirm – By Index	목록에서 확인 객체를 순서에 따라 참조합니다. GetObjectByIndex 메소드를 사용해야 합니다.
3	String	GetObjectStr	LastMessage	최신 에러 정보
1	Event	RegEvent	OnNewConfirm	새로운 정정확인 및 취소확인이 발생하면 이를 통보합니다.

GxConfirm

정정 및 취소 주문의 확인에 대한 정보를 담고 있는 객체입니다.

ID	변수유형	사용가능 메소드	필드명	설명
1	Numeric	GetObject	Order	정정 및 취소 확인 대상 주문
2	Numeric	GetObjectVal	ArrivedTime	주문 확인 (PC)도착 시간
3	Numeric	GetObjectVal	ReqQty	주문 요청 수량
4	Numeric	GetObjectVal	ConfirmQty	확인 수량, 확인수량이 0 이면 확인이 거부된 것입니다.
5	Numeric	GetObjectVal	ConfirmPrice	확인 가격(정정 주문에 대한 확인만 해당, 정정주문의 실제 정정가임)
6	Numeric	GetObjectVal	NearPrice	정정가격 유형(정정 주문에 대한 확인만 해당) 0 : 지정가 / 1 : 시장가 2 : 조건부지정가 / 3 : 최유리지정가
7	String	GetObjectStr	RjtReason	확인 거부 사유(주문이 정정되거나 취소 되기전 체결 등으로 사라질 경우 등) 확인 거부 시만 값을 갖습니다.
8	String	GetObjectStr	LastMessage	최신 에러 내용

GxOrderHandler

주문을 내고, 주문의 거래소 접수까지 상태를 알려주는 객체입니다.

ID	변수유형	사용가능 메소드	필드명	설명
1	Numeric	GetObject	Order	주문 요청 목록 객체
2	Numeric	GetObjectVal	WaitCount	서버로 보내기 전 상태에 있는 주문요청 건수입니다.
3	String	GetObjectStr	LastMessage	최종 에러 메시지

GxOrderReqs

주문 요청 목록을 관리하는 객체입니다.

ID	변수유형	사용가능 메소드	필드명	설명
3	Numeric	GetObjectVal	Count	총 주문 요청 건수. 이미 접수가 끝난 것과 아직 서버에 보내지 않은 것들이 함께 포함된 것입니다.
0	Numeric	GetObjectByIndex	GxOrderReq	주문 요청 내용을 목록에서 순서에 따라 참조합니다.
4	String	GetObjectStr	LastMessage	최신 에러 메시지

GxOrderReq

건별 주문 요청 내용을 담고 있는 객체입니다.

ID	변수유형	사용가능 메소드	필드명	설명
1	Numeric	GetObject	Account	주문 대상 계좌 객체
2	Numeric	GetObject	Symbol	주문 대상 종목 객체
3	Numeric	GetObjectVal	PositionType	매수/매도 구문
4	Numeric	GetObjectVal	Qty	주문 수량
5	Numeric	GetObjectVal	PriceType	주문 가격 유형
6	Numeric	GetObjectVal	Price	주문 가격
7	Numeric	GetObject	TargetOrder	정정 및 취소 주문 요청의 경우 정정 및 취소 대상 주문 객체
8	Numeric	GetObjectVal	OrderType	주문 유형
9	Numeric	GetObjectVal	State	주문 요청 상태
10	Numeric	GetObjectVal	SrvAcptNo	서버 접수 번호. 서버 접수 시 값을 갖습니다.
11	String	GetObjectStr	RjtReason	주문 거부 사유(전송거부, 서버거부, 증전거부 시)
12	Bool	GetObjectBool	IsDone	처리가 종료여부(전송거부, 서버거부, 증전거부, 증전접수)
13	Bool	GetObjectBool	IsDoing	처리중 여부(신규, 전송대기, 서버접수대기, 증전접수대기)
14	String	GetObjectStr	LastMessage	최신 에러 정보

GxEmHandler

서버 체결 통보 지연 등 서버 장애 발생시 고수의 엔진을 거치지 않고 서버에 직접 주문을 전송하는데 사용합니다.

ID	변수유형	사용가능 메소드	필드명	설명
1	String	GetObjectStr	LastMessage	최신 에러 메시지
1	Event	RegEvent	OnEmOrderArrived	서버 주문 접수 결과를 통보합니다. GxEmHandler 에서는 주문 전송 및 접수가 고수 엔진을 거치지 않으므로 주문 접수 결과 통보시 계좌, 종목 등 상세 주문 정보가 제공이 되지 않습니다. 사용하실 때 유의하시기 바랍니다.

GxSymbolStore

종목을 종합적으로 관리합니다. 종목을 참조할 수 있는 다양한 방법을 제공하므로 편리하게 종목 정보를 조회할 수 있습니다.

ID	변수유형	사용가능 메소드	필드명	설명
3	Numeric	GetObjectVal	Count	전체 종목 수(분류되지 않은 전체 종목 목록입니다.)
0	Numeric	GetObjectByIndex GetObjectByStr	GxSymbol – By Index	종목을 전체 목록에서 순서 및 코드에 따라 조회합니다. GetObjectByIndex, GetObjectByStr 메소드를 이용해야합니다.
4	String	GetObjectStr	LastMessage	최신 에러 정보
5	Numeric	GetObject	Stocks	시가 총액 상위 5 개 주식 목록만 따로 분류해 놓은 것입니다. (객체)
6	Numeric	GetObject	Futures	KOSPI200 지수 선물만 따로 분류해 놓은 것입니다. (객체)
7	Numeric	GetObject	Spreads	KOSPI200 지수 선물 스프레드 종목만 분류해 놓은 것입니다. (객체)
8	Numeric	GetObject	OptionMonths	KOSPI200 지수 옵션을 월물별로 분류해 놓은 것입니다. (객체)
9	Numeric	GetObject	AllFutures	KOSPI200 지수선물, 국내주식선물, 국내선물 객체
10	Numeric	GetObject	BondFutures	국내선물 중 국채선물 객체
11	Numeric	GetObject	CurrencyFutures	국내선물 중 통화선물 객체
12	Numeric	GetObject	ProductFutures	국내선물 중 상품선물 객체
13	Numeric	GetObject	K200Index	KOSPI200 지수 객체
14	Numeric	GetObject	NearestFuture	선물 최근월물 객체
15	Numeric	GetObject	NearestOptMonth	옵션 최근월 참조 객체
16	Bool	GetObjectBool	QuoteEventFiltered	일정 시간 간격으로 호가 이벤트를 발생시킬 것인지를 결정하거나 현재 설정 되어 있는 값을 읽습니다.
17	Numeric	GetObjectVal	QuoteEventInterval	QuoteEventFiltered 를 사용할 경우 그 시간 간격을 설정하거나 현재 설정 되어 있는 값을 읽습니다.
18	Bool	GetObjectBool	PriceEventFiltered	일정 시간 간격으로 현재가 이벤트를 발생시킬 것인지를 결정하거나 현재 설정 되어 있는 값을 읽습니다.
19	Numeric	GetObjectVal	PriceEventInterval	PriceEventFiltered 를 사용할 경우 그 시간 간격을 설정하거나 현재 설정되어 있는 값을

				읽습니다.
20	Bool	GetObjectBool	TickEventFiltered	체결 거르기(이전 전달된 체결과 동일 호가 이면서 같은 분봉 이내 이면 거래량을 누적하고 체결을 구독 객체로 보내지 않음)을 사용할 것인지를 결정하거나 설정되어 있는 값을 읽습니다.
21	Numeric	GetObjectVal	TickEventInterval	TickEventFiltered 를 사용할 경우 필요한 '최대 대기 시간' 값을 설정하거나 그 값을 읽습니다.

GxSymbol

종목 목록을 관리합니다.

ID	변수유형	사용가능 메소드	필드명	설명
1	String	GetObjectStr	Code	종목 코드
2	String	GetObjectStr	Desc	종목명(축약)
3	String	GetObjectStr	KorDesc	한글 종목명
4	String	GetObjectStr	EngDesc	영문 종목명
5	Numeric	GetObjectVal	UnderlyingType	기초자산 종류 0 : KOSPI200 / 1 : KOSDAQ50 2 : 주식
6	Numeric	GetObjectVal	SymbolType	상품종류 0 : 지수타입 / 1 : 주식타입 2 : 선물타입 / 3 : 옵션타입 4 : 스프레드타입 / 5 : 채권타입
7	Numeric	GetObjectVal	OptionType	옵션 종류(옵션만 해당) 0 : Call 종목 / 1 : Put 종목
8	Numeric	GetObjectVal	StrikePrice	행사가(옵션만 해당)
9	String	GetObjectStr	ClosingMonth	만기월일(형식: "YYYYMMDD")
10	Numeric	GetObjectVal	MatureDate	만기일
11	Numeric	GetObjectVal	TrailingDays	잔존일수
12	Numeric	GetObjectVal	PointValue	1.0 포인트당 가격(원)
13	Numeric	GetObjectVal	CDRate	CD 금리
14	Numeric	GetObjectVal	HighLimit	상한가
15	Numeric	GetObjectVal	LowLimit	하한가
16	Numeric	GetObjectVal	CBHigh	CB 상한가
17	Numeric	GetObjectVal	CBLow	CB 하한가
18	Numeric	GetObjectVal	BasePrice	기준가

19	Numeric	GetObjectValEx	LongThValues	매수 구간 이론가 목록
20	Numeric	GetObjectValEx	ShortThValues	매도 구간 이론가 목록
21	Numeric	GetObjectValEx	VLongThValues	매수 구간 이론가
22	Numeric	GetObjectValEx	VShortThValues	매도 구간 이론가
23	Numeric	GetObjectVal	Precision	가격 표시 단위(소수점 자리수)
24	Bool	GetObjectBool	IsNearest	근월물 여부
25	Bool	GetObjectBool	IsATM	ATM 여부
26	Numeric	GetObjectVal	IV	종목 내재변동성(옵션만 해당) – 이 값은 증권거래소에서 장시작전 일회 제공하는 값입니다.
27	Numeric	GetObjectVal	PrevOpenInterest	전일 미결제 약정 수량
28	Numeric	GetObjectVal	Open	시가
29	Numeric	GetObjectVal	High	고가
30	Numeric	GetObjectVal	Low	저가
31	Numeric	GetObjectVal	Close	현재가
32	Numeric	GetObjectVal	Change	현재가 전일비
33	Numeric	GetObjectVal	OpenInterest	종목 미결제약정
34	Numeric	GetObjectVal	AccVolume	누적 거래량
35	Numeric	GetObjectVal	AccAmount	누적 거래대금
36	Numeric	GetObject	Quote	5 단계 호가 객체
37	Numeric	GetObject	LastTick	최종 체결 정보 객체
38	Bool	GetObjectBool	CanOrder	주문가능여부
39	String	GetObjectStr	LastMessage	최신 에러 메시지
40	Numeric	GetObjectVal	ThPrice	이론가(선물, 옵션만 해당)
41	Numeric	GetObjectVal	IV2	종목 내재변동성(옵션만 해당) 입니다. IV2 는 IV 와는 달리 고수에서 제공하는 내재 변동성 계산식에 의한 값입니다.
42	Numeric	GetObjectVal	Delta	Delta(옵션만 해당)
43	Numeric	GetObjectVal	Gamma	Gamma(옵션만 해당)
44	Numeric	GetObjectVal	Theta	Theta(옵션만 해당)
45	Numeric	GetObjectVal	Vega	Vega(옵션만 해당)
46	Numeric	GetObjectVal	Lamda	Lamda(옵션만 해당)
47	Numeric	GetObjectVal	IntrinsicValue	옵션 내재 가치(옵션만 해당)
48	Numeric	GetObjectVal	TimeValue	시간 가치(옵션만 해당)
49	Numeric	GetObjectVal	EventOmitCount	최근 PriceEventFiltered 를 사용함으로써 생략된 이벤트 수, PriceEventFiltered 를 사용한 효율을

				알 수 있습니다.
1	Event	RegEvent	OnPriceChanged	종목의 현재가가 변할 때 발생합니다.

GxQoute

GxSymbol 의 하위 객체로 GxSymbol 의 종목 정보 중 호가 정보를 이 객체에 위임하여 관리합니다.

종목의 5 단계 호가 정보를 클라이언트에 제공하며 클라이언트에 호가 이벤트를 발생합니다.

ID	변수유형	사용가능 메소드	필드명	설명
1	Numeric	GetObjectVal	UpdateTime	호가 최종 갱신 시간
2	Numeric	GetObjectValEx	Prices	5 단계 호가, 단 원래 수치에 100 이 곱해진 값이므로 실제 사용할 때는 100 으로 나누어야 함 0~4 매도 호가 5~9 매수 호가
3	Numeric	GetObjectValEx	Qtys	5 단계 호가 잔량 0 : 매도 총 잔량 1~5 : 매도 잔량
4	Numeric	GetObjectValEx	Cnts	5 단계 호가 건수 0 : 매도 총 건수 1~5 : 매도 건수 6 : 매수 총 건수 7~11 매수 건수
5	Numeric	GetObjectValEx	Quotes	5 단계 호가/잔량/건수를 한번에 조회 0~4 : 매도호가 5~9 : 매수호가 10 : 매도 총 잔량 11~15 : 매도잔량 16 : 매수 총 잔량 17~21 : 매수잔량 22 : 매도 총 건수 23~27 : 매도건수 28 : 매수 총 건수 29~33 : 매수건수
6	Numeric	GetObjectValEx	VPrices	5 단계 호가, 원래 수치에 100 이 곱해짐
7	Numeric	GetObjectValEx	VQtys	5 단계 호가 잔량
8	Numeric	GetObjectValEx	VCnts	5 단계 호가 건수
9	Numeric	GetObjectValEx	VQuotes	5 단계 호가/잔량/건수를 한번에 조회
13	String	GetObjectStr	LastMessage	최신 에러 정보

14	Numeric	GetObjectVal	EventOmitCount	최근 QuoteEventFiltered 를 사용함으로써 생략된 이벤트 수, QuoteEventFiltered 를 사용한 효율을 알 수 있습니다.
1	Events	RegEvent	OnQuoteChanged	종목의 호가가 변할 때 발생합니다.

GxLastTick

종목의 최종 체결에 대한 정보를 담고 있습니다.

ID	변수유형	사용가능 메소드	필드명	설명
1	Numeric	GetObjectVal	Time	최종 체결 시각
2	Numeric	GetObjectVal	LSType	매수체결/매도체결 구분
3	Numeric	GetObjectVal	Price	체결가
4	Numeric	GetObjectVal	Volume	체결량
5	Numeric	GetObjectVal	AccVolume	누적체결량
6	Numeric	GetObjectVal	OpenInterest	종목 미결제약정
7	String	GetObjectStr	LastMessage	최신 에러 메시지
1	Event	RegEvent	OnNewTick	종목의 체결이 들어올 때 발생합니다.

GxSymbols

종목 목록을 관리합니다.

ID	변수유형	사용가능 메소드	필드명	설명
3	Numeric	GetObject	Count	종목 수
0	Numeric	GetObjectByIndex GetObjectByStr	GxSymbol – By Index	종목을 목록에서 순서 및 코드에 따라 조회합니다. GetObjectByIndex, GetObjectByStr 메소드를 이용해야합니다.
4	String	GetObjectStr	LastMessage	최신 에러 메시지

GxOptionMonths

옵션의 월물 목록을 관리합니다.

ID	변수유형	사용가능 메소드	필드명	설명
3	Numeric	GetObjectVal	Count	옵션월물 수
0	Numeric	GetObjectByIndex	GxOptionMonth – By Index	옵션월물을 목록에서 순서에 따라 조회합니다. GetObjectByIndex 메소드를 이용해야 합니다.
4	String	GetObjectStr	LastMessage	최신 에러 정보

GxOptionMonth

옵션 월물에 대한 정보를 담고 있습니다.

ID	변수유형	사용가능 메소드	필드명	설명
1	String	GetObjectStr	MonthCode	옵션 월물의 코드입니다. (형식 : "YYYYMMDD")
2	String	GetObjectStr	Item	옵션 월물의 설명입니다. (형식 : "MM 월")
3	Numeric	GetObject	StrikePrices	옵션월물의 행사가를 조회합니다. 옵션월물의 종목들을 조회하고자 할 때는 행사가를 통해서 조회를 합니다.
4	String	GetObjectStr	LastMessage	최신 에러 정보

GxStrikePrices

옵션 월물의 행사가 목록을 관리합니다.

ID	변수유형	사용가능 메소드	필드명	설명
3	Numeric	GetObjectVal	Count	행사가 수
0	Numeric	GetObjectByIndex	GxStrikePrice – By Index	행사가를 목록에서 순서에 따라 조회합니다. GetObjectByIndex 메소드를 이용해야 합니다.
4	String	GetObjectStr	LastMessage	최신 에러 정보

GxStrikePrice

행사가에 대한 정보를 담고 있습니다.

ID	변수유형	사용가능 메소드	필드명	설명
1	Numeric	GetObjectVal	StrikePrice	행사가
2	String	GetObjectStr	StrikeDesc	행사가 설명 (형식 : '127.5', '127.0')
3	Numeric	GetObject	Call	해당 행사가의 콜 종목
4	Numeric	GetObject	Put	해당 행사가의 풋 종목
5	String	GetObjectStr	LastMessage	최신 에러 정보

GxServerInfo

증권사 서버 연결 정보를 조회하고, 서버와 연결이 끊어졌을 경우 이를 알려주는 역할을 합니다.

ID	변수유형	사용가능 메소드	필드명	설명
1	Bool	GetObjectBool	Connected	고수와 증권사 서버와의 연결 여부를 알려줍니다. 연결되어 있으면 값이 True 가 됩니다.
2	Numeric	GetObjectVal	ServerDate	서버 날짜를 조회합니다.
3	String	GetObjectStr	LastMessage	최신 에러 정보

(2) 메소드

GetObject

문법
GetObject(대상객체, Field ID)

대상객체의 Field ID 에 해당하는 객체를 반환합니다. 여기서 대상객체, Field ID 모두 Numeric 변수입니다. 모든 GOM 객체는 고유한 Numeric 값을 갖고 있으며, GetObject 메소드를 통해 GOM 객체에 접근이 가능합니다.

GOM 구조의 최상위객체인 GxServer 의 고유 Numeric 값은 0 입니다. 만약 GxServer 객체의 3 번 필드의 객체에 접근하고자 할 경우 아래와 같이 실행하면 됩니다.

```
Object(NoReAlloc): oGom(DLL("STage_Root", "GosuGOM.dll", ""))
GxTradeStore = oGom.GetObject( 0, 3 )
```

또는 아래와 같이 변수를 지정해서 할 수도 있습니다.

```
Object(NoReAlloc): oGom(DLL("STage_Root", "GosuGOM.dll", ""))
Var: GxServer(0), GxTradeStore(0)
GxTradeStore = oGom.GetObject( GxServer, 3 )
```

GxServer 객체의 Field ID 3 번은 GxTradeStore 객체이며, 여기서 GetObject 메소드는 GxTradeStore 객체의 Numeric 값을 반환하여 GxTradeStore 에 저장합니다.

한가지 예를 더 살펴보겠습니다. 만약 GxAccounts 객체에 접근하고자 한다면, GxServer -> GxTradeStore -> GxAccounts 순서로 접근하면 됩니다. GxTradeStore 는 GxServer 의 필드 ID 3 번이며, GxAccounts 는 GxTradeStore 의 필드 ID 1 번이므로 아래와 같이 실행하면 됩니다.

```
Object(NoReAlloc): oGom(DLL("STage_Root", "GosuGOM.dll", ""))
Var: GxServer(0), GxTradeStore(0), GxAccounts(0)

GxTradeStore = oGom.GetObject( GxServer, 3 )
GxAccounts = oGom.GetObject( GxTradeStore, 1 )
```

GetObjectByIndex

문법

GetObjectByIndex(대상객체, Field ID, Index 순서)

GOM 객체에서 **GxAccounts, GxPositions, GxOrders, GxFills, GxConfirms, GxOrderHandler, GxOrderReqs, GxSymbolStore, GxOptionMonths, GxStrikePrices, GxSymbols** 는 Collection 객체입니다. 예를 들어 GxPositions 의 경우 포지션 목록을 관리하는 객체이며 하위 객체로 GxPosition 이 있습니다. 아래 그림과 같은 구조이며, 이러한 하위객체에 접근하고자 할 때는 Index 또는 문자열로 접근할 수 있습니다. GetObjectByIndex 는 Index 를 통해 하위객체에 접근하게 할 수 있는 메소드입니다.

GxPositions (Collection Object)						
1	2	3	4	5	…	N
GxPosition	GxPosition	GxPosition	GxPosition	GxPosition	…	GxPosition

GxPositions 의 경우 Fields ID 0 번을 통해 아래와 같이 GxPosition 에 접근할 수 있습니다.

```
Object(NoReAlloc): oGom(DLL("STage_Root", "GosuGOM.dll", ""))
Var: GxServer(0), GxTradeStore(0), GxPositions(0), GxPosition(0)

GxTradeStore = oGom.GetObject( GxServer, 3 )
GxPositions = oGom.GetObject( GxTradeStore, 2 )
GxPosition = oGom.GetObjectByIndex( GxPositions, 0, 1 )
```

GetObjectByIndex(GxPositions, 0, 1)에서 1 은 Index 순서로 첫번째 GxPosition 을 의미합니다.

GetObjectByStr

문법

GetObjectByStr(대상객체, Field ID, 코드값)

GOM 객체에서 **GxAccounts, GxSymbolStore, GxSymbols** 는 GetObjectByIndex 뿐만 아니라 GetObjectByStr 을 통해 문자열로 접근이 가능합니다. 계좌번호 또는 종목명을 문자열로 입력하여 하위객체에 접근할 수 있습니다.

아래의 예는 GxAccounts 의 경우 계좌번호가 "99999000001"인 하위객체에 접근하는 방법입니다.

```
Object(NoReAlloc): oGom(DLL("STage_Root", "GosuGOM.dll", ""))
Var: GxServer(0), GxTradeStore(0), GxAccounts(0), GxAccount(0)

GxTradeStore = oGom.GetObject( GxServer, 3 )
GxAccounts = oGom.GetObject( GxTradeStore, 1 )
GxAccount = oGom.GetObjectByStr( GxAccounts, 0, "99999000001" )
```

GetObjectVal

문법

GetObjectVal(대상객체, Field ID)

대상객체의 Field ID 에 해당하는 Numeric 값을 반환합니다. 여기서 대상객체, Field ID 모두 Numeric 변수입니다.

GxAccounts 객체의 경우 Field ID 3 번은 Count 로 총 계좌수를 나타냅니다. 아래와 같이 GetObjectVal(GxAccounts, 3) 을 통해 Count 값을 불러올 수 있습니다.

```
Object(NoReAlloc): oGom(DLL("STage_Root", "GosuGOM.dll", ""))
Var: GxServer(0), GxTradeStore(0), GxAccounts(0), Count(0)

GxTradeStore = oGom.GetObject( GxServer, 3 )
GxAccounts = oGom.GetObject( GxTradeStore, 1 )
Count = GetObjectVal( GxAccounts, 3 )
```

GetObjectStr

문법

GetObjectStr(대상객체, Field ID)

대상객체의 Field ID 에 해당하는 String 값을 반환합니다. 여기서 대상객체, Field ID 모두 Numeric 변수입니다.

GxServer 의 Field ID 1 번은 LastMessage 로 최신 에러 정보를 갖고 있습니다. 아래와 같이 GetObjectStr(GxServer, 1) 을 통해 LastMessage 값을 불러올 수 있습니다.

```
Object(NoReAlloc): oGom(DLL("STage_Root", "GosuGOM.dll", ""))
Var: GxServer(0), LastMessage("")

LastMessage = oGom.GetObjectStr( GxServer, 1 )
```

GetObjectBool

문법

GetObjectBool(대상객체, Field ID)

대상객체의 Field ID 에 해당하는 Boolean 값을 반환합니다. 여기서 대상객체, Field ID 모두 Numeric 변수입니다.

GxAccount 의 Field ID 4 번은 Authenticated 로 비밀번호 확인 여부를 갖고 있습니다. (TRUE / FALSE) GxServer->GxTradeStore->GxAccounts->GxAccount 순으로 접근하여 GetObjectBool 을 이용해 Authenticated 를 가져올 수 있습니다.

```
Object(NoReAlloc): oGom(DLL("STage_Root", "GosuGOM.dll", ""))
Var: GxServer(0), GxTradeStore(0), GxAccounts(0), GxAccount(0), Auth(FALSE)

GxTradeStore = oGom.GetObject( GxServer, 3 )
GxAccounts = oGom.GetObject( GxTradeStore, 1 )
GxAccount = oGom.GetObjectByIndex( GxAccounts, 0, 1 )
Auth = oGom.GetObjectBool( GxAccount, 4 )
```

RegEvent

문법

RegEvent(대상객체, Field ID, 이벤트 갱신간격)

이벤트 객체를 등록합니다. 대상객체의 Field ID 에 해당하는 이벤트 객체를 등록하며, 이벤트 갱신간격을 ms 단위로 지정합니다. 이벤트 갱신간격을 0 으로 할 경우, 객체의 특정 속성이 변할 경우 실시간으로 이벤트를 발생시킵니다.

GOM 객체 중 GxAccounts, GxPositions, GxOrders, GxFills, GxConfirms, GxSymbol, GxQuote, GxLastTick 객체에는 이벤트 객체가 있습니다. 이벤트 객체는 객체의 특정 속성이 변할 경우 이벤트를 발생시키며, 이벤트의 발생 유무에 따라 스테이지 전략을 구성할 수 있습니다.

이벤트를 사용하기 위해서는 아래와 같이 RegEvent 로 해당 이벤트를 등록하여야 합니다.

```
Object(NoReAlloc): oGom(DLL("STage_Root", "GosuGOM.dll", ""))
Var: GxServer(0), GxTradeStore(0), GxAccounts(0)

GxTradeStore = oGom.GetObject( GxServer, 3 )
GxAccounts = oGom.GetObject( GxTradeStore, 1 )
oGom.RegEvent( GxAccounts, 1, 0 )
```

활용예제는 GOM 활용예제의 3) 계좌정보 불러오기 – 계좌정보 가져오기 응용 – 이벤트 활용예제를 살펴보시기 바랍니다.

UnregEvent

문법

UnregEvent(대상객체, Field ID)

RegEvent 로 등록한 이벤트를 해제합니다.

GOM 객체 중 GxAccounts, GxPositions, GxOrders, GxFills, GxConfirms, GxSymbol, GxQuote, GxLastTick 객체에는 이벤트 객체가 있습니다. 이벤트 객체는 객체의 특정 속성이 변할 경우 이벤트를 발생시키며, 이벤트의 발생 유무에 따라 스테이지 전략을 구성할 수 있습니다. 이벤트 객체의 등록은 RegEvent 로 가능하며 UnregEvent 는 등록한 이벤트를 해제합니다.

아래와 같이 등록한 이벤트를 해제할 수 있습니다.

```
Object(NoReAlloc): oGom(DLL("STage_Root", "GosuGOM.dll", ""))
Var: GxServer(0), GxTradeStore(0), GxAccounts(0)

GxTradeStore = oGom.GetObject( GxServer, 3 )
GxAccounts = oGom.GetObject( GxTradeStore, 1 )
oGom.RegEvent( GxAccounts, 1, 0 )

oGom.UnregEvent( GxAccounts, 1 )
```

추가 활용예제는 GOM 활용예제의 3) 계좌정보 불러오기 – 계좌정보 가져오기 응용 – 이벤트 활용예제를 살펴보시기 바랍니다.

EventObject

문법

EventObject

RegEvent 로 대상객체에 이벤트를 등록한 상황에서 이벤트가 발생할 경우, EventObject 는 이벤트가 속해있는 객체를 가리킵니다.

아래는 GxAccounts 의 Field ID 1 번 이벤트를 등록한 것입니다. GxAccounts 의 Field ID 1 번 이벤트는 OnAccountUpdated 로 계좌의 속성값(예탁금, 증거금, 손익, 수수료 등)이 변경될 때 발생합니다. 계좌의 속성 변경으로 이벤트가 발생할 때, oGom.EventObject 는 GxAccounts 를 가리킵니다.

```
Object(NoReAlloc): oGom(DLL("STage_Root", "GosuGOM.dll", ""))
Var: GxServer(0), GxTradeStore(0), GxAccounts(0)

GxTradeStore = oGom.GetObject( GxServer, 3 )
GxAccounts = oGom.GetObject( GxTradeStore, 1 )
oGom.RegEvent( GxAccounts, 1, 0 )

If oGom.EventObject = GxAccounts And oGom.Event = 1 Then
    Value1 = oGom.GetObjectVal( oGom.TarObject, 6 )
End If
```

추가 활용예제는 GOM 활용예제의 3) 계좌정보 불러오기 – 계좌정보 가져오기 응용 – 이벤트 활용예제를 살펴보시기 바랍니다.

Event

문법

Event

RegEvent 로 대상객체에 이벤트를 등록한 상황에서 이벤트가 발생할 경우, Event 는 이벤트가 속해있는 객체의 이벤트 Field ID 를 가리킵니다.

아래는 GxAccounts 의 Field ID 1 번 이벤트를 등록한 것입니다. GxAccounts 의 Field ID 1 번 이벤트는 OnAccountUpdated 로 계좌의 속성값(예탁금, 증거금, 손익, 수수료 등)이 변경될 때 발생합니다. 계좌의 속성 변경으로 이벤트가 발생할 때, oGom.Event 는 Field ID 인 1 을 가리킵니다.

```
Object(NoReAlloc): oGom(DLL("STage_Root", "GosuGOM.dll", ""))
Var: GxServer(0), GxTradeStore(0), GxAccounts(0)

GxTradeStore = oGom.GetObject( GxServer, 3 )
GxAccounts = oGom.GetObject( GxTradeStore, 1 )
oGom.RegEvent( GxAccounts, 1, 0 )

If oGom.EventObject = GxAccounts And oGom.Event = 1 Then
    Value1 = oGom.GetObjectVal( oGom.TarObject, 6 )
End If
```

추가 활용예제는 GOM 활용예제의 3) 계좌정보 불러오기 – 계좌정보 가져오기 응용 – 이벤트 활용예제를 살펴보시기 바랍니다.

TarObject

문법
TarObject

RegEvent 로 대상객체에 이벤트를 등록한 상황에서 이벤트가 발생할 경우, TarObject 는 이벤트를 발생시킨 해당객체에 접근할 수 있게 합니다.

아래는 GxAccounts 의 Field ID 1 번 이벤트를 등록한 것입니다. GxAccounts 의 Field ID 1 번 이벤트는 OnAccountUpdated 로 계좌의 속성값(예탁금, 증거금, 손익, 수수료 등)이 변경될 때 발생합니다. 계좌의 속성 변경으로 이벤트가 발생할 때, oGom.TarObject 는 이벤트를 발생시킨 계좌를 가리킵니다. 이 경우 계좌객체는 GxAccounts 의 하위객체인 GxAccount 가 됩니다.

```
Object(NoReAlloc): oGom(DLL("STage_Root", "GosuGOM.dll", ""))
Var: GxServer(0), GxTradeStore(0), GxAccounts(0)

GxTradeStore = oGom.GetObject( GxServer, 3 )
GxAccounts = oGom.GetObject( GxTradeStore, 1 )
oGom.RegEvent( GxAccounts, 1, 0 )

If oGom.EventObject = GxAccounts And oGom.Event = 1 Then
    Value1 = oGom.GetObjectVal( oGom.TarObject, 6 )
End If
```

추가 활용예제는 GOM 활용예제의 3) 계좌정보 불러오기 – 계좌정보 가져오기 응용 – 이벤트 활용예제를 살펴보시기 바랍니다.

GetObjectValEx

문법

GetObjectValEx(대상객체, Index, Field ID)

GxQuote 객체의 경우 Field ID 하나에 여러 정보를 담고 있습니다. 예를 들어 Field ID 2 번의 경우 5 단계 호가정보를 갖고 있으며, 매도호가 5 단계와 매수호가 5 단계의 정보를 갖고 있습니다. 총 10 개의 정보를 갖고 있으며 이에 접근할 때는 0~9 의 Index 숫자를 사용합니다.

0~4 는 매도호가, 5~9 는 매수호가이므로 아래와 같이 접근할 수 있습니다.

GetObjectValEx(GxQuote, 0, 2)	매도호가1 * 100	GetObjectValEx(GxQuote, 5, 2)	매수호가1 * 100
GetObjectValEx(GxQuote, 1, 2)	매도호가2 * 100	GetObjectValEx(GxQuote, 6, 2)	매수호가2 * 100
GetObjectValEx(GxQuote, 2, 2)	매도호가3 * 100	GetObjectValEx(GxQuote, 7, 2)	매수호가3 * 100
GetObjectValEx(GxQuote, 3, 2)	매도호가4 * 100	GetObjectValEx(GxQuote, 8, 2)	매수호가4 * 100
GetObjectValEx(GxQuote, 4, 2)	매도호가5 * 100	GetObjectValEx(GxQuote, 9, 2)	매수호가5 * 100

아래는 GxQuote 의 Field ID 2 번 호가정보에 접근하는 방법입니다.

```
Object(NoReAlloc): oGom(DLL("STage_Root", "GosuGOM.dll", ""))
Var: GxServer(0), GxSymbolStore(0), GxSymbol(0), GxQuote(0)
GxSymbolStore = oGom.GetObject( GxServer, 4 )
GxSymbol = oGom.GetObjectByStr( GxSymbolStore, 0, "101F9" )
GxQuote = oGom.GetObject( GxSymbol, 36 )

SellPrice1 = oGom.GetObjectValEx( GxQuote, 0, 2 ) / 100
SellPrice2 = oGom.GetObjectValEx( GxQuote, 1, 2 ) / 100
SellPrice3 = oGom.GetObjectValEx( GxQuote, 2, 2 ) / 100
SellPrice4 = oGom.GetObjectValEx( GxQuote, 3, 2 ) / 100
SellPrice5 = oGom.GetObjectValEx( GxQuote, 4, 2 ) / 100

BuyPrice1 = oGom.GetObjectValEx( GxQuote, 5, 2 ) / 100
BuyPrice2 = oGom.GetObjectValEx( GxQuote, 6, 2 ) / 100
BuyPrice3 = oGom.GetObjectValEx( GxQuote, 7, 2 ) / 100
BuyPrice4 = oGom.GetObjectValEx( GxQuote, 8, 2 ) / 100
BuyPrice5 = oGom.GetObjectValEx( GxQuote, 9, 2 ) / 100
```

RunObject

문법

RunObject(대상객체, Field ID)

GOM 객체 중 메소드를 제공하는 GOM 객체가 있습니다. GxTradeStore 의 경우 Field ID 9 번 Synchronize 라는 메소드가 있습니다. (이 메소드는 고수의 주문/체결 내역 수동 조회 기능을 요청하는 역할을 합니다.) 이러한 GOM 객체의 메소드를 사용할 수 있는 STage 의 메소드가 RunObject 입니다.

RunObject 는 GOM 메소드 중 리턴값이 존재하지 않는 메소드에 접근할 때 사용합니다. 리턴값이 존재하는 경우에는 RunObjectVal 을 사용합니다.

RunObject 및 RunObjectVal 을 사용하기에 앞서 GOM 메소드에 입력변수를 전달해야 하는데 이를 위해 AddParamVal, AddParamStr, AddParamBool 을 사용합니다.

다음은 주문과 관련한 RunObject 의 사용예 입니다.

```
oGom.AddParamStr("99999000001")
oGom.AddParamStr("101F9")
oGom.AddParamVal(0)
oGom.AddParamVal(1)
oGom.AddParamVal(1)
oGom.AddParamVal(300)

vRtn = oGom.RunObjectVal(GxOrderHandler, 4)
oGom.RunObject(GxOrderHandler,7)
```

RunObjectVal

문법

RunObjectVal(대상객체, Field ID)

GOM 객체의 메소드에 접근할 때 사용합니다. RunObjectVal 은 GOM 메소드 중 리턴값이 존재하는 메소드에 접근할 때 사용합니다.

RunObjectVal 을 사용하기에 앞서 GOM 메소드에 입력변수를 전달해야 하는데 이를 위해 AddParamVal, AddParamStr, AddParamBool 을 사용합니다.

다음은 주문과 관련한 RunObject 의 사용예 입니다.

```
oGom.AddParamStr("99999000001")
oGom.AddParamStr("101F9")
oGom.AddParamVal(0)
oGom.AddParamVal(1)
oGom.AddParamVal(1)
oGom.AddParamVal(300)

vRtn = oGom.RunObjectVal(GxOrderHandler, 4)
oGom.RunObject(GxOrderHandler,7)
```

AddParamStr

문법

AddParamStr(String)

GOM 객체의 메소드에 접근할 때 사용합니다. RunObject 및 RunObjectVal 을 사용하기에 앞서 GOM 메소드에 입력변수를 전달해야 하는데 이를 위해 AddParamVal, AddParamStr, AddParamBool 을 사용합니다. AddParamStr 은 문자열 변수를 전달할 때 사용합니다.

다음은 주문과 관련한 RunObject 의 사용예 입니다.

```
oGom.AddParamStr("99999000001")
oGom.AddParamStr("101F9")
oGom.AddParamVal(0)
oGom.AddParamVal(1)
oGom.AddParamVal(1)
oGom.AddParamVal(300)

vRtn = oGom.RunObjectVal(GxOrderHandler, 4)
oGom.RunObject(GxOrderHandler,7)
```

AddParamVal

문법

AddParamVal(Numeric)

GOM 객체의 메소드에 접근할 때 사용합니다. RunObject 및 RunObjectVal 을 사용하기에 앞서 GOM 메소드에 입력변수를 전달해야 하는데 이를 위해 AddParamVal, AddParamStr, AddParamBool 을 사용합니다. AddParamVal 은 숫자열 변수를 전달할 때 사용합니다.

다음은 주문과 관련한 RunObject 의 사용예 입니다.

```
oGom.AddParamStr("99999000001")
oGom.AddParamStr("101F9")
oGom.AddParamVal(0)
oGom.AddParamVal(1)
oGom.AddParamVal(1)
oGom.AddParamVal(300)

vRtn = oGom.RunObjectVal(GxOrderHandler, 4)
oGom.RunObject(GxOrderHandler,7)
```

AddParamBool

문법

AddParamBool(Boolean)

GOM 객체의 메소드에 접근할 때 사용합니다. RunObject 및 RunObjectVal 을 사용하기에 앞서 GOM 메소드에 입력변수를 전달해야 하는데 이를 위해 AddParamVal, AddParamStr, AddParamBool 을 사용합니다. AddParamBool 은 Boolean 변수를 전달할 때 사용합니다.

```
oGom.AddParamBool( True )
```

GOM 메소드 사용하기

STage 의 RunObject 및 AddParam 등을 이용하여 GOM 메소드를 사용할 수 있습니다. GOM 에는 아래 같은 메소드들이 있습니다.

GxTradeStore

ID	메소드	설명
9	Synchronize	고수의 주문/체결 내역 수동 조회 기능을 요청합니다. 주문 접수 및 체결이 예상보다 늦어질 때는 이 기능을 사용하여 강제적으로 최신 정보를 갱신합니다. 이 메소드는 최소한 사용 빈도를 줄여야 하며 고수에서도 1 초에 최대 한번 밖에 기능을 하지 않습니다. 그래서 만약 이 메소드가 5 번 호출한다면 실제 기능은 한번만 사용되며 고수에서 주문이 나간 후 5 초 후에는 자동으로 고수에서 이 기능을 호출하므로 이 후에 강제 조회를 하실 필요가 없습니다. 하루종일 1 초에 한번씩 이 메소드를 호출하는 등의 행위는 피하시고 가급적으로 꼭 필요한 시점에서만 호출하십시오. 이 메소드의 무리한 호출은 주문에 안좋은 영향을 줄 수 있습니다.
	사용예	**oGom.RunObject(GxTradeStore, 9)**

GxOrders

ID	메소드	설명
1	FindOrder	계좌코드, 종목코드, 주문접수번호로 주문을 찾습니다. **Parameters** **String 계좌코드 / String 종목코드 / Numeric 주문접수번호** **Return** 해당되는 주문이 있으면 해당주문객체(Numeric), 없으면 Null 객체, 포지션이 없으면 Null
	사용예	**oGom.AddParamStr("99999000001")** **oGom.AddParamStr("101F9")** **oGom.AddParamVal(1400)** **vRtn = oGom.RunObjectVal(GxOrders, 1)**

GxPositions

ID	메소드	설명
1	FindPosition	계좌코드, 종목코드로 포지션 객체를 찾습니다. **Parameters** **String 계좌코드 / String 종목코드** **Return** 찾는 포지션이 있을 경우 포지션객체(Numeric), 없으면 Null
	사용예	**oGom.AddParamStr("99999000001")** **oGom.AddParamStr("101F9")** **vRtn = oGom.RunObjectVal(GxPositions, 1)**

GxAccounts

ID	메소드	설명
1	FindAccount	계좌이름으로 계좌객체를 찾습니다. **Parameters** **String 계좌명** **Return** 계좌객체(Numeric)
	사용예	**oGom.AddParamStr("㈜델타익스체인지")** **vRtn = oGom.RunObjectVal(GxAccounts, 1)**

GxAccount

ID	메소드	설명
1	ForceUpdate	예탁금, 증거금, 수수료 정보를 서버로부터 새로 수신을 합니다. 고수에서는 예탁금, 증거금, 수수료를 자동으로 최신정보로 갱신을 합니다. 하지만 서버에서의 처리 지연 등으로 실제와 달라질 수 있으므로, 이때 서버에서 최신정보를 받아 오도록 할 때 이 메소드를 사용합니다.
	사용예	**oGom.RunObject(GxAccount, 1)**

GxOrderHandler

ID	메소드	설명
4	PutNewOrder	신규 주문을 주문요청 목록에 추가합니다. 이때 주문은 서버에 전송되는 것이 아닙니다. 주문 요청 목록을 만드신 후 Send()를 호출할 때, 실제 서버로 주문이 전달됩니다. **Parameters** **String 계좌코드** **String 종목코드** **Numeric 매수, 매도구분** 0 : 매수 / 1 : 매도 **Numeric 가격유형** 0 : 지정가 / 1 : 시장가 / 2 : 조건부지정가 / 3 : 최유리 지정가 **Numeric 주문수량** **Numeric 주문가격** **Return** 주문요청객체(Numeric, GxOrderReq). 차후 주문의 접수 여부를 확인할 때 사용합니다. 주문 요청에 문제가 있을 경우에는 Null 값이 됩니다.
	사용예	**oGom.AddParamStr("99999000001")** **oGom.AddParamStr("101F9")** **oGom.AddParamVal(0)** **oGom.AddParamVal(1)** **oGom.AddParamVal(1)** **oGom.AddParamVal(300)** **vRtn = oGom.RunObjectVal(GxOrderHandler, 4)**
5	PutChangeOrder	정정 주문을 주문요청 목록에 추가합니다. **Parameters** **Numeric 정정 대상 주문객체(GxOrder)** **Numeric 정정수량** **Numeric 가격유형** 0 : 지정가 / 1 : 시장가 / 2 : 조건부지정가 / 3 : 최유리 지정가

		Numeric 정정가격 **Return** 주문요청객체(Numeric, GxOrderReq). 차후 주문의 접수 여부를 확인할 때 사용합니다. 주문 요청에 문제가 있을 경우에는 Null 값이 됩니다.
	사용예	**oGom.AddParamVal(GxOrder)** **oGom.AddParamVal(1)** **oGom.AddParamVal(1)** **oGom.AddParamVal(280.00)** **vRtn = oGom.RunObjectVal(GxOrderHandler, 5)**
6	PutCancelOrder	취소 주문을 주문요청 목록에 추가합니다. **Parameters** **Numeric 정정 대상 주문객체(GxOrder)** **Numeric 취소 수량** **Return** 주문요청객체(Numeric, GxOrderReq). 차후 주문의 접수 여부를 확인할 때 사용합니다. 주문 요청에 문제가 있을 경우에는 Null 값이 됩니다.
	사용예	**oGom.AddParamVal(GxOrder)** **oGom.AddParamVal(5)** **vRtn = oGom.RunObjectVal(GxOrderHandler, 6)**
7	Send	주문 요청 목록에 새로 추가된 주문들을 실제 서버로 전송합니다.
	사용예	**oGom.RunObject(GxOrderHandler,7)**
8	PutNewOrder2	신규 주문을 주문요청 목록에 추가합니다. (IOC, FOK 주문가능) **Paremeters** **String 계좌코드** **String 종목코드** **Numeric 매수, 매도구분** 0 : 매수 / 1 : 매도

		Numeric 가격유형 0 : 지정가 / 1 : 시장가 / 2 : 조건부지정가 / 3 : 최유리 지정가 **Numeric 주문수량** **Numeric 주문가격** **Numeric 체결유형(일반/IOC/FOK)** 0 : FOK 주문 / 1 : IOC 주문 / 2 : 일반주문 **Return** 주문요청객체(Numeric, GxOrderReq). 차후 주문의 접수 여부를 확인할 때 사용합니다. 주문 요청에 문제가 있을 경우에는 Null 값이 됩니다.
	사용예	**oGom.AddParamStr("99999000001")** **oGom.AddParamStr("101F9")** **oGom.AddParamVal(0)** **oGom.AddParamVal(1)** **oGom.AddParamVal(1)** **oGom.AddParamVal(300)** **oGom.AddParamVal(1)** **vRtn = oGom.RunObjectVal(GxOrderHandler, 8)**
9	PutChangeOrder2	정정 주문을 주문요청 목록에 추가합니다. **Parameters** **Numeric 정정 대상 주문객체(GxOrder)** **Numeric 정정수량** **Numeric 가격유형** 0 : 지정가 / 1 : 시장가 / 2 : 조건부지정가 / 3 : 최유리 지정가 **Numeric 정정가격** **Numeric 체결유형(일반/IOC/FOK)** 0 : FOK 주문 / 1 : IOC 주문 / 2 : 일반주문 **Return** 주문요청객체(Numeric, GxOrderReq). 차후 주문의 접수 여부를 확인할 때 사용합니다. 주문 요청에 문제가 있을 경우에는 Null 값이 됩니다.
	사용예	**oGom.AddParamVal(GxOrder)** **oGom.AddParamVal(1)**

		oGom.AddParamVal(1) **oGom.AddParamVal(280.00)** **oGom.AddParamVal(1)** **vRtn = oGom.RunObjectVal(GxOrderHandler, 9)**

GxEmHandler

ID	메소드	설명
2	SendNewOrder	신규 주문을 냅니다. **Parameters** **String 계좌코드** **String 종목코드** **Numeric 매수,매도구분** 0 : 매수 / 1 : 매도 **Numeric 가격유형** 0 : 지정가 / 1 : 시장가 / 2 : 조건부지정가 / 3 : 최유리 지정가 **Numeric 주문수량** **Numeric 주문가격** **Return** 주문요청객체(Numeric, GxOrderReq). 차후 주문의 접수 여부를 확인할 때 사용합니다. 주문 요청에 문제가 있을 경우에는 Null 값이 됩니다.
	사용예	**oGom.AddParamStr("99999000001")** **oGom.AddParamStr("101F9")** **oGom.AddParamVal(0)** **oGom.AddParamVal(1)** **oGom.AddParamVal(1)** **oGom.AddParamVal(300)** **vRtn = oGom.RunObjectVal(GxEmhandler, 2)**
3	SendChangeOrder	정정 주문을 냅니다. **Parameters** **Numeric 정정 대상 주문객체(GxOrder)**

		Numeric 정정수량 **Numeric 가격유형** 0 : 지정가 / 1 : 시장가 / 2 : 조건부지정가 / 3 : 최유리 지정가 **Numeric 정정가격** **Return** 주문요청객체(Numeric, GxOrderReq). 차후 주문의 접수 여부를 확인할 때 사용합니다. 주문 요청에 문제가 있을 경우에는 Null 값이 됩니다.
	사용예	**oGom.AddParamVal(GxOrder)** **oGom.AddParamVal(1)** **oGom.AddParamVal(1)** **oGom.AddParamVal(280.00)** **vRtn = oGom.RunObjectVal(GxEmHandler, 3)**
4	SendCancelOrder	취소 주문을 냅니다. **Parameters** **Numeric 정정 대상 주문객체(GxOrder)** **Numeric 취소 수량** **Return** 주문요청객체(Numeric, GxOrderReq). 차후 주문의 접수 여부를 확인할 때 사용합니다. 주문 요청에 문제가 있을 경우에는 Null 값이 됩니다.
	사용예	**oGom.AddParamVal(GxOrder)** **oGom.AddParamVal(5)** **vRtn = oGom.RunObjectVal(GxEmHandler, 4)**
5	SendNewOrder2	신규 주문을 냅니다. **Paremeters** **String 계좌코드** **String 종목코드** **Numeric 매수, 매도구분** 0 : 매수 / 1 : 매도 **Numeric 가격유형**

		0 : 지정가 / 1 : 시장가 / 2 : 조건부지정가 / 3 : 최유리 지정가 **Numeric 주문수량** **Numeric 주문가격** **Numeric 체결유형(일반/IOC/FOK)** 0 : FOK 주문 / 1 : IOC 주문 / 2 : 일반주문 **Return** 주문요청객체(Numeric, GxOrderReq). 차후 주문의 접수 여부를 확인할 때 사용합니다. 주문 요청에 문제가 있을 경우에는 Null 값이 됩니다.
	사용예	**oGom.AddParamStr("99999000001")** **oGom.AddParamStr("101F9")** **oGom.AddParamVal(0)** **oGom.AddParamVal(1)** **oGom.AddParamVal(1)** **oGom.AddParamVal(300)** **oGom.AddParamVal(1)** **vRtn = oGom.RunObjectVal(GxEmHandler, 5)**
6	SendChangeOrder2	정정 주문을 냅니다. **Parameters** **Numeric 정정 대상 주문객체(GxOrder)** **Numeric 정정수량** **Numeric 가격유형** 0 : 지정가 / 1 : 시장가 / 2 : 조건부지정가 / 3 : 최유리 지정가 **Numeric 정정가격** **Numeric 체결유형(일반/IOC/FOK)** 0 : FOK 주문 / 1 : IOC 주문 / 2 : 일반주문 **Return** 주문요청객체(Numeric, GxOrderReq). 차후 주문의 접수 여부를 확인할 때 사용합니다. 주문 요청에 문제가 있을 경우에는 Null 값이 됩니다.
	사용예	**oGom.AddParamVal(GxOrder)** **oGom.AddParamVal(1)** **oGom.AddParamVal(1)** **oGom.AddParamVal(280.00)**

oGom.AddParamVal(1) **vRtn = oGom.RunObjectVal(GxEmHandler, 6)**

GxSymbolStore

ID	메소드	설명
1	SetGreeksParam	GxSymbol 의 Greeks 계산에 필요한 값(잔존일수 계산방식, Call, Put 입력 변동성값, 기초자산 선물비율 등을 설정합니다. 이 설정을 하지 않는 경우 Greeks 값들은 기본값인 전일 대표 내재 변동성, 달력날짜, 선물비율=0 인 값으로 계산됩니다. **Parameters** **Numeric 잔존일수 계산방식** 0 : 달력일수 날짜기준 1 : 달력일수 시간기준 2 : 거래일수 날짜기준 3 : 거래일수 시간기준 **Numeric 기초자산 선물 비율** 0 ~ 1 사이의 실수, 0.0 이면 KOSPI200 만 반영, 1.0 이면 선물만 반영 **Numeric 그릭스 계산에 필요한 Call 입력 변동성 값 (1 ~ 300)** -100 입력시 해당 설정 무시 -1 입력시 종목 내재 변동성 사용 -2 입력시 현재 ATM 변동성 사용 -3 입력시 현재 대표 변동성 사용 -4 입력시 전일 대표 변동성 사용 -15 입력시 역사적 변동성 15 일 사용 -30 입력시 역사적 변동성 30 일 사용 -40 입력시 역사적 변동성 40 일 사용 -60 입력시 역사적 변동성 60 일 사용 -90 입력시 역사적 변동성 90 일 사용 **Numeric 그릭스 계산에 필요한 Put 입력 변동성 값 (1 ~ 300)** 특정한 변동성 사용은 Call 의 설정과 동일합니다.
사용예		**oGom.AddParamVal()**

	oGom.AddParamVal(0.8) **oGom.AddParamVal(-2)** **oGom.AddParamVal(-2)** **oGom.RunObject(GxSymbolStore, 1)**

GxQuote

ID	메소드	설명
10	Price	5 단 호가 중 개별호가를 조회합니다. **Parameters** **Numeric 매수, 매도 구분** 0 : 매수 / 1 : 매도 **Numeric 호가단계 (1 ~ 5)** **Return** 해당 호가(Numeric)
	사용예	**oGom.AddParamVal(0)** **oGom.AddParamVal(1)** **vRtn = oGom.RunObjectVal(GxQuote, 10)**
11	Qty	5 단 호가 중 개별호가잔량을 조회합니다. **Parameters** **Numeric 매수, 매도 구분** 0 : 매수 / 1 : 매도 **Numeric 호가단계 (0 : 총잔량, 1 ~ 5)** **Return** 해당 호가(Numeric)
	사용예	**oGom.AddParamVal(0)** **oGom.AddParamVal(1)** **vRtn = oGom.RunObjectVal(GxQuote, 11)**
12	Cnt	5 단 호가 중 개별호가건수를 조회합니다. **Parameters**

	Numeric 매수, 매도 구분 0 : 매수 / 1 : 매도 **Numeric 호가단계 (1 ~ 5)** **Return** 해당 호가(Numeric)
사용예	**oGom.AddParamVal(0)** **oGom.AddParamVal(1)** **vRtn = oGom.RunObjectVal(GxQuote, 12)**

5. GOM 의 활용예제

STage 에서 GOM 을 사용하고자 지표를 작성할 때 실행코드는 크게 3 부분으로 구분 됩니다.

(1) STage 에 GOM 연결하기
(2) 변수선언 및 초기화
(3) GOM 객체 정보 및 메소드에 접근하여 원하는 기능 구현

(1) STage 에 GOM 연결하기

```
Object(NoReAlloc): oGom(DLL("STage_Root", "GosuGOM.dll", ""))
```

앞서 설명 드린 바 있듯이 GOM 을 STage 에서 사용할 수 있도록 연결하는 구문입니다. GOM 을 스테이지에서 사용하기 위해서는 지표에 위의 구문을 반드시 포함시켜야 합니다. oGom 오브젝트를 생성하게 되며, oGom.GetObject( ), oGom.GetObjectVal( ) 등의 메소드 사용이 가능하게 됩니다. 이를 통해 GOM 객체에 접근할 수 있습니다.

Object(NoReAlloc) :	오브젝트를 선언 및 생성합니다. 기존 Object: 선언의 경우, 선언 이후 new 명령어를 통해 오브젝트 생성을 하는 방식이었습니다. Object(NoReAlloc): 선언은 선언과 동시에 오브젝트를 생성하며, 이후 new 를 통한 동일 오브젝트의 재생성은 불가능합니다. NoReAlloc 은 No Re-Allocation 의 의미로 오브젝트의 반복 생성을 막게 됩니다. GOM 오브젝트 생성은 STage 와 GOM 연결을 위한 것으로 반복 생성이 필요 없으며, 혹시라도 있을지 모를 반복 생성으로 인한 비효율을 막기 위해 NoReAlloc 선언을 사용합니다.
oGom	GOM 을 연결하기 위한 오브젝트 이름입니다. 임의로 지정이 가능하나, 명확한 의미 전달을 위해 이후 모든 예제에서는 oGom 으로 사용합니다. (object_Gom, GOM 오브젝트라는 의미)
DLL	DLL 은 새로운 오브젝트 타입입니다. 기존 오브젝트로는 Line, Text, Rectangle 등이 있습니다.

DLL(“GOM 의 경로”, “GOM 의 파일명”, “전달인자”)	DLL( )은 DLL 오브젝트를 생성하는 함수입니다. 3 가지 변수를 입력 받습니다. 첫 번째 변수는 GOM DLL 파일의 경로이며, 두 번째 변수는 GOM DLL 파일의 파일명입니다. 세 번째 변수는 전달인자입니다. **DLL(“STage_Root”, “GosuGOM.dll”, “”)** STage_Root 는 STage 폴더를 가리키는 예약어입니다. GosuGOM.dll 은 GOM DLL 파일의 파일명이며, 세 번째 변수는 “”으로 현재는 전달인자가 없습니다. 이와 같이 DLL 함수를 호출하면 GOM 오브젝트를 리턴하게 됩니다.

(2) 변수선언 및 초기화

STage 에서 내부변수 선언은 Var: 를 통해 이루어집니다. (Var, Vars, Variable, Variables, Param, Params, Parameter, Parameters 모두 동의어입니다.) Var: 선언에는 **NoRecover** 와 **NoReInit** 의 두 가지 옵션을 추가할 수 있으며 **GOM 오브젝트 관련 변수는 Var(NoRecover, NoReInit)의 선언 사용을 필수적으로 해야합니다.**

NoRecover
NoRecover 옵션을 통해 선언된 내부변수는 실시간 기준으로 사용할 수 있도록 구현된 것입니다.

NoReInit
지표의 적용과 동시에 변수에 값이 할당되도록 구현된 것입니다.

다음의 예제는 Var:선언과 Var(NoRecover):, Var(NoReInit):의 차이를 비교하기 위한 예제입니다. 예제를 통해 NoRecover 와 NoReInit 을 살펴보겠습니다.

```
Var:Count1(0)
Var(NoRecover):Count2(0)
Var(NoReInit):Count3(0)

k = C + C[5]

Count1 = Count1 + 1
Count2 = Count2 + 1
Count3 = Count3 + 1

plot1(Count1, "Count1")
plot2(Count2, "Count2")
plot3(Count3, "Count3")
```

Var:Count1(0)
Var(NoRecover):Count2(0)
Var(NoReInit):Count3(0)

Count1 은 NoRecover 와 NoReInit 옵션이 적용되지 않은 변수이며, Count2 는 NoRecover 옵션이 적용된 변수입니다. Count3 은 NoReInit 옵션이 적용되었습니다.

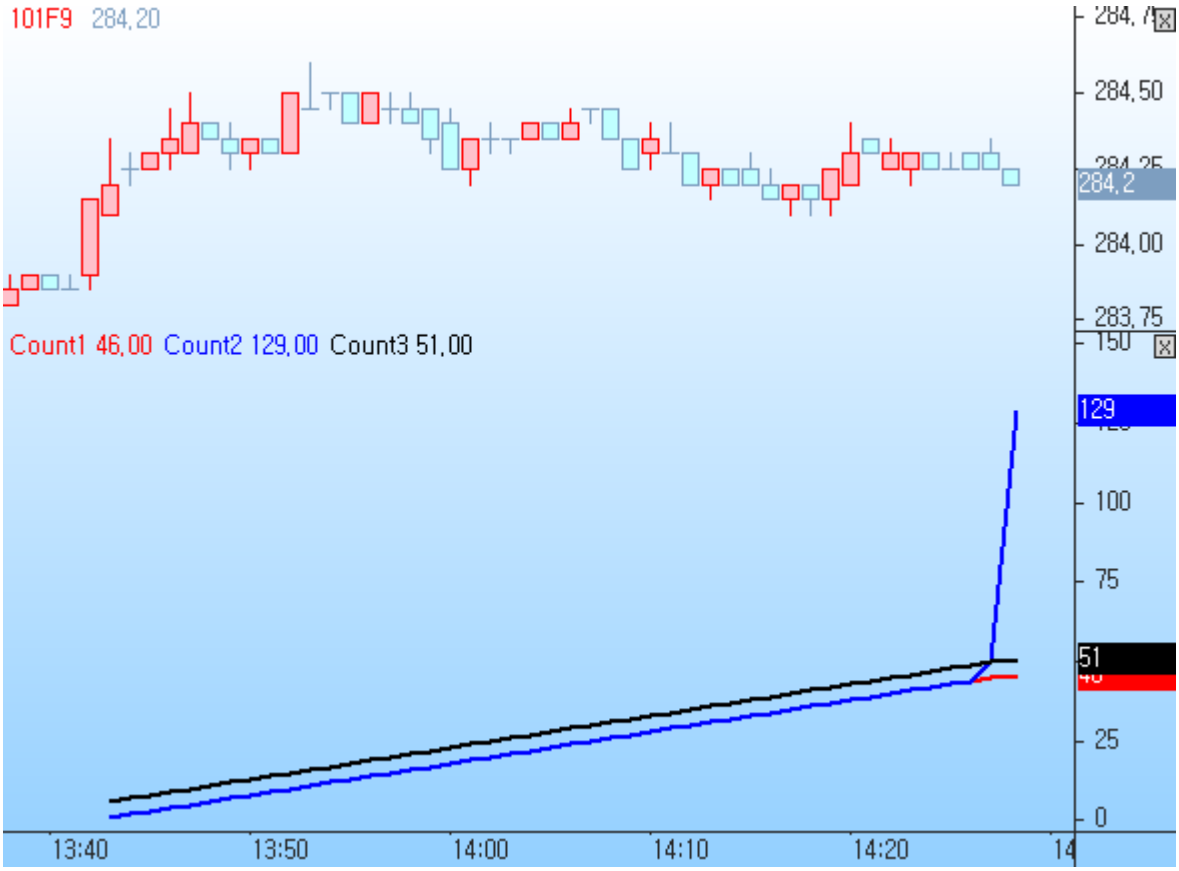
k = C + C[5]

이 부분은 NoReInit 의 차이를 보이기위해 포함된 구문입니다. k 라는 변수에 현재봉의 종가와 5 번째 이전봉의 종가를 저장합니다.

Count1 = Count1 + 1
Count2 = Count2 + 1
Count3 = Count3 + 1

Count1, Count2, Count3 의 값을 1 만큼 증가시킵니다.

위의 예제를 STage 차트에서 실행하면 다음과 같은 화면이 나옵니다.



Count1 = 46 Count2 = 129 Count3 = 51
위와 같이 세 변수의 값은 모두 차이를 나타냅니다.

우선 Count1 와 Count3 의 차이부터 살펴보겠습니다. 두 변수의 차이는 **NoReInit** 옵션으로 선언이 되었는지 여부입니다. Count3 의 경우 **NoReInit** 옵션이 사용되었습니다.

예제의 내용 중에 **k = C + C[5]** 라는 부분이 있습니다. k 는 현재봉의 5 번째 이전봉을 참조하기 때문에 k 값이 지정되기 위해서는 최소한 6 개의 봉 데이터가 필요합니다. 스테이지에서는 이러한 경우 6 번째 봉 이후부터 전체 지표가 수행되게 됩니다. Count1 = Count1 + 1 은 6 번째 봉부터 수행되게 됩니다. 만약 지표 내에 30 일 이동평균선이 사용되었다면, 30 개의 봉 참조가 필요하기 때문에 차트의 31 번째봉부터 전체 지표가 수행될 수 있게됩니다.

반면 NoReInit 옵션으로 선언된 Count3 는 이와상관없이 곧바로 명령이 수행되도록 구현되었습니다.

Count2 는 **NoRecover** 옵션으로 선언이되었습니다.

스테이지의 지표 및 전략은 봉단위로 적용이 되며, 내부변수 또한 이에 맞게 구현되어 있습니다. Count1 = Count1 + 1 의 경우 1 만큼의 값 증가가 봉단위로 이루어 지게 됩니다.

하지만 상황에 따라서는 봉주기와 관계 없이 틱단위로 내부변수 값이 저장되어야 할 수 있습니다. 특히 GOM 을 사용하게 되면 실시간으로 값과 이벤트가 업데이트 되기 때문에, 이를 실시간으로 저장해야할 필요가 있습니다.

NoRecover 옵션으로 변수를 선언하면 실시간 기준으로 변수를 사용할 수 있습니다. Count2 는 NoRecover 로 선언되었기 때문에 틱단위로 실행이 되었으며 위와 같은 값을 갖게 된 것입니다.

TradeStation 의 경우 IntrabarPersist 타입을 통해 이와 같은 틱단위 변수를 지원하고 있습니다. 스테이지의 NoRecover 는 이와 동일한 기능을 제공합니다.

위에서 살펴본바와 같이 GOM 의 사용은 과거 봉단위 시스템 트레이딩이 아닌 실시간 트레이딩의 적용을 의미하며, 그렇기 때문에 GOM 관련 변수는 모두 Vars(NoRecover, NoReInit)로 선언되어야 합니다.

(3) GOM 객체 정보 및 메소드에 접근하여 원하는 기능 구현

STage 에 GOM 연결하기와 변수선언 및 초기화가 완료되고 나면 STage-GOM 메소드를 사용하여 GOM 객체에 접근할 수 있습니다. 활용 방법은 다음의 예제들을 살펴보시길 바랍니다.

< GxTradeStore >

1) 계좌정보 불러오기 – 계좌번호와 계좌이름 가져오기

```
// 계좌번호와 계좌이름 가져오기
Object(NoReAlloc): oGom(DLL("STage_Root", "GosuGOM.dll", ""))          [1]
Var(NoRecover, NoReInit): xbFirst(True),
        GxServer(0), GxTradeStore(0), GxAccounts(0), GxAccount(0),     [2]
        Count(0), Code(""), Desc("")

If xbFirst Then

        GxTradeStore = oGom.GetObject(GxServer, 3)
        GxAccounts = oGom.GetObject(GxTradeStore, 1)
        Count = oGom.GetObjectVal(GxAccounts, 3)

        GxAccount = oGom.GetObjectByIndex(GxAccounts, 0, 1)            [3]

        Code = oGom.GetObjectStr(GxAccount, 1)
        Desc = oGom.GetObjectStr(GxAccount, 2)

        xbFirst = False

End If
```

[1] Object(NoReAlloc): oGom(DLL("STage_Root", "GosuGOM.dll", ""))

GOM 을 스테이지에서 사용하기 위해서는 지표에 위의 구문을 반드시 포함시켜야 합니다.
oGom 오브젝트를 생성하게 되며, oGom.GetObject( ), oGom.GetObjectVal( ) 등의 메소드 사용이 가능하게 됩니다. 이를 통해 GOM 객체에 접근할 수 있습니다.

[2] Var(NoRecover, NoReInit): xbFirst(True), GxServer(0), GxTradeStore(0), GxAccounts(0), GxAccount(0), Count(0), Code(""), Desc("")

GOM 을 사용하는 과정에서 필요한 변수를 선언하는 구문입니다. GxServer, GxTradeStore, GxAccounts, GxAccount 는 GOM 의 해당객체에 대응시킬 변수입니다. Count, Code, Desc 는 객체 정보를 저장할 변수입니다.

[3]

```
GxTradeStore = oGom.GetObject(GxServer, 3)
GxAccounts = oGom.GetObject(GxTradeStore, 1)
Count = oGom.GetObjectVal(GxAccounts, 3)
GxAccount = oGom.GetObjectByIndex(GxAccounts, 0, 1)
Code = oGom.GetObjectStr(GxAccount, 1)
Desc = oGom.GetObjectStr(GxAccount, 2)
```

여기서 등장하는 메소드는 GetObject( ), GetObjectByIndex( ), GetObjectVal( ), GetObjectStr( ) 입니다. 이를 이용하여 GOM 객체에 접근할 수 있습니다.

GxTradeStore = oGom.GetObject(GxServer, 3)

GxServer 객체의 Field ID 3 번인 GxTradeStore 객체에 접근하여 객체의 고유 Numeric 값을 GxTradeStore 변수에 저장합니다.

GxAccounts = oGom.GetObject(GxTradeStore, 1)

GxTradeStore 객체의 Field ID 1 번인 GxAccounts 객체에 접근하여 객체의 고유 Numeric 값을 GxAccounts 변수에 저장합니다.

Count = oGom.GetObjectVal(GxAccounts, 3)

GxAccounts 객체의 Field ID 3 번은 총 계좌 수를 나타냅니다. 이를 Count 변수에 저장합니다. 숫자열 값이므로 GetObjectVal 을 사용합니다.

GxAccount = oGom.GetObjectByIndex(GxAccounts, 0, 1)

GetObjectByIndex 메소드를 이용하여 Collection Object 인 GxAccounts 로부터 첫번째 계좌객체에 접근합니다. 계좌객체의 고유 Numeric 값을 GxAccount 변수에 저장합니다.

Code = oGom.GetObjectStr(GxAccount, 1)
Desc = oGom.GetObjectStr(GxAccount, 2)

GxAccount 객체의 Field ID 1 번과 2 번은 각각 계좌번호와 계좌명을 나타냅니다. 이를 Code, Desc 변수에 저장합니다. 문자열 값이므로 GetObjectStr 을 사용합니다.

If xbFirst Then ~~~~ End If

[3]과 같이 GOM 객체 값을 불러오는 과정을 If 문을 이용해 지표 실행 시 1 번만 실행되게 한 것입니다. xbFirst 변수를 True 로 초기화하여 선언한 뒤 [GOM 객체 값 불러오기]를 완료하면 xbFirst = False 로 하여 IF 문을 빠져나가게 합니다. 이와 같은 식으로 값 불러오기가 한번만 이루어질수록 할 수 있습니다.

이 경우 xbFirst 변수는 Var:가 아닌 Var(NoRecever, NoReInit):으로 선언 및 초기화 되어야 합니다.

2) 계좌정보 불러오기 – 기타 계좌정보 가져오기

```
//예탁금총액, 증거금총액, 잠정수수료, 당일 총 손익 가져오기
Object(NoReAlloc): oGom(DLL("STage_Root", "GosuGOM.dll", ""))
Var(NoRecover, NoReInit): xbFirst(True),
        GxServer(0), GxTradeStore(0), GxAccounts(0), GxAccount(0),
        DepositTotal(0), MarginTotal(0), Commission(0), DailyPL(0)

If xbFirst Then

        GxTradeStore = oGom.GetObject(GxServer, 3)
        GxAccounts = oGom.GetObject(GxTradeStore, 1)
        GxAccount = oGom.GetObjectByIndex(GxAccounts, 0, 1)

        DepositTotal = oGom.GetObjectVal(GxAccount, 6)
        MarginTotal = oGom.GetObjectVal(GxAccount, 8)
        Commission = oGom.GetObjectVal(GxAccount, 12)
        DailyPL = oGom.GetObjectVal(GxAccount, 13)

        xbFirst = False

End If
```

이전 예제에서는 계좌번호와 계좌명을 가져왔습니다. 이번에는 예탁금총액, 증거금총액, 잠정수수료, 당일 총 손익을 조회하였습니다.

```
DepositTotal = oGom.GetObjectVal(GxAccount, 6)
MarginTotal = oGom.GetObjectVal(GxAccount, 8)
Commission = oGom.GetObjectVal(GxAccount, 12)
DailyPL = oGom.GetObjectVal(GxAccount, 13)
```

위와 같이 GetObjectVal 을 이용하여 원하는 정보에 접근할 수 있습니다. 해당하는 Field ID 를 전달하면 됩니다.

3) 계좌정보 불러오기 – 계좌정보 가져오기 응용 – 이벤트 활용예제

```
//이벤트를 활용하여 계좌정보 실시간 갱신하기
Object(NoReAlloc): oGom(DLL("STage_Root", "GosuGOM.dll", ""))
Var(NoRecover, NoReInit): xbFirst(True),
        GxServer(0), GxTradeStore(0), GxAccounts(0), GxAccount(0),
        DepositTotal(0), MarginTotal(0), Commission(0), DailyPL(0)

If xbFirst Then

        GxTradeStore = oGom.GetObject(GxServer, 3)
        GxAccounts = oGom.GetObject(GxTradeStore, 1)
        GxAccount = oGom.GetObjectByIndex(GxAccounts, 0, 1)

        DepositTotal = oGom.GetObjectVal(GxAccount, 6)
        MarginTotal = oGom.GetObjectVal(GxAccount, 8)
        Commission = oGom.GetObjectVal(GxAccount, 12)
        DailyPL = oGom.GetObjectVal(GxAccount, 13)

        oGom.RegEvent(GxAccounts, 1, 0)

        xbFirst = False

End If

If oGom.EventObject = GxAccounts And oGom.Event = 1 Then

        DepositTotal = oGom.GetObjectVal(oGom.TarObject, 6)
        MarginTotal = oGom.GetObjectVal(oGom.TarObject, 8)
        Commission = oGom.GetObjectVal(oGom.TarObject, 12)
        DailyPL = oGom.GetObjectVal(oGom.TarObject, 13)

End If
```

이번 예제에서는 RegEvent 의 사용이 추가되었습니다. RegEvent 는 이벤트를 등록하는 메소드입니다. GxAccounts 의 Field ID 1 번 이벤트는 OnAccountUpdated 로 계좌의 속성값이 변경될 때 이벤트가 발생합니다.

```
oGom.RegEvent(GxAccounts, 1, 0)
```

위와 같이 GxAccounts 의 1 번 이벤트를 등록할 수 있습니다. 위에서 세 번째 입력변수인 0 은 이벤트 갱신간격으로 여기서 0 은 0ms, 즉 실시간을 의미합니다. 이렇게 등록된 이벤트는 EventObject 와 Event 를 이용한 IF 문을 통해 활용할 수 있습니다.

```
If oGom.EventObject = GxAccounts And oGom.Event = 1 Then

    DepositTotal = oGom.GetObjectVal(oGom.TarObject, 6)
    MarginTotal = oGom.GetObjectVal(oGom.TarObject, 8)
    Commission = oGom.GetObjectVal(oGom.TarObject, 12)
    DailyPL = oGom.GetObjectVal(oGom.TarObject, 13)

End If
```

이벤트가 발생하게 되면 EventObject 와 Event 에 이벤트가 속해있는 객체와 Field ID 가 저장되게 됩니다. 위와 같이 If 문을 통해 해당이벤트가 발생했을 때 If 문 안의 내용을 실행하도록 할 수 있습니다. 여기서는 DepositTotal, MarginTotal, Commission, DailyPL 변수를 갱신합니다.

```
DepositTotal = oGom.GetObjectVal(oGom.TarObject, 6)
```

위의 구문에서 TarObject 는 이벤트가 발생한 객체에 접근하도록 합니다. 이 경우 이벤트는 계좌의 속성값이 변경될 때 발생하기 때문에, 속성값이 변한 계좌 객체가 TarObject 가 됩니다. 위와 같은 방법으로 실시간으로 변화하는 객체 정보를 업데이트 할 수 있습니다.

4) 포지션정보 불러오기 – 포지션 정보 가져오기

```
//가장 최근 포지션의 수량과 평가손익 가져오기
Object(NoReAlloc): oGom(DLL("STage_Root", "GosuGOM.dll", ""))
Var(NoRecover, NoReInit): xbFirst(True),
        GxServer(0), GxTradeStore(0), GxPositions(0), GxPosition(0),
        Count(0), Qty(0), EvalPL(0)

If xbFirst Then

        GxTradeStore = oGom.GetObject(GxServer, 3)
        GxPositions = oGom.GetObject(GxTradeStore, 2)
        Count = oGom.GetObjectVal(GxPositions, 3)

        GxPosition = oGom.GetObjectByIndex(GxPositions, 0, Count)
        Qty = oGom.GetObjectVal(GxPosition, 3)
        EvalPL = oGom.GetObjectVal(GxPosition, 7)

        xbFirst = False

End If
```

이번 예제는 GxPositions 객체에서 포지션 정보를 가져오는 예제입니다.

GxPositions = oGom.GetObject(GxTradeStore, 2)

GxTradeStore 의 Field ID 2 번을 통해 GxPositions 객체에 접근합니다.

Count = oGom.GetObjectVal(GxPositions, 3)

GxPositions 의 Field ID 3 번은 포지션 수를 나타냅니다. 이를 Count 에 저장합니다.

GxPosition = oGom.GetObjectByIndex(GxPositions, 0, Count)
Qty = oGom.GetObjectVal(GxPosition, 3)
EvalPL = oGom.GetObjectVal(GxPosition, 7)

GxPositions 는 Collection Object 입니다. 아래와 같은 식의 하위구조를 갖고 있으며, 가장 최근의 포지션에 접근하기 위해서는 GetObjectByIndex 를 이용하여 접근하면 됩니다.

GxPositions (Collection Object)						
1	2	3	4	5	...	N
GxPosition	GxPosition	GxPosition	GxPosition	GxPosition	...	GxPosition

GxPosition = oGom.GetObjectByIndex(GxPositions, 0, Count)

위의 구문을 통해 마지막 포지션 객체에 접근할 수 있습니다.

이후 GxPositions 에서 Field ID 3 번과 Field ID 7 번에 해당하는 최종 포지션 수량과 평가손익을 가져와 Qty 와 EvalPL 에 저장합니다.

5) 주문목록 정보 불러오기 – 주문목록

```
//가장 최근 주문목록의 매수/매도 타입과 수량을 가져오기
Object(NoReAlloc): oGom(DLL("STage_Root", "GosuGOM.dll", ""))
Var(NoRecover, NoReInit): xbFirst(True),
        GxServer(0), GxTradeStore(0), GxOrders(0), GxOrder(0),

        Count(0),
        PositionType(0), Qty(0)

If xbFirst Then

        GxTradeStore = oGom.GetObject(GxServer, 3)
        GxOrders = oGom.GetObject(GxTradeStore, 3)
        Count = oGom.GetObjectVal(GxOrders, 3)

        GxOrder = oGom.GetObjectByIndex(GxOrders, 0, Count)
        PositionType = oGom.GetObjectVal(GxOrder, 5)
        Qty = oGom.GetObjectVal(GxOrder, 7)

        xbFirst = False

End If
```

앞서 GxPositions 의 경우와 유사합니다. 위의 굵은 글씨로 표시된 부분을 통해 PositionType 과 Qty 변수에 매수/매도 타입과 수량을 저장할 수 있습니다. PositionType 의 경우 0 은 매수주문을 1 은 매도주문을 의미합니다.

6) 체결목록 정보 불러오기

```
//가장 최근 체결목록의 거래소 체결번호와 체결수량을 가져오기
Object(NoReAlloc): oGom(DLL("STage_Root", "GosuGOM.dll", ""))
Var(NoRecover, NoReInit): xbFirst(True),
        GxServer(0), GxTradeStore(0), GxFills(0), GxFill(0),

        Count(0),
        FillNo(0), FillQty(0)

If xbFirst Then

        GxTradeStore = oGom.GetObject(GxServer, 3)
        GxFills = oGom.GetObject(GxTradeStore, 4)
        Count = oGom.GetObjectVal(GxFills, 3)

        GxFill = oGom.GetObjectByIndex(GxFills, 0, Count)
        FillNo = oGom.GetObjectVal(GxFill, 2)
        FillQty = oGom.GetObjectVal(GxFill, 3)

        xbFirst = False

End If
```

위의 구문을 통해 FillNo 와 FillQty 에 거래소 체결번호와 체결수량을 가져올 수 있습니다.

7) 정정/취소 확인 목록 정보 불러오기

```
//가장 최근의 정정/취소 확인 목록의 확인수량과 확인가격 가져오기
Object(NoReAlloc): oGom(DLL("STage_Root", "GosuGOM.dll", ""))
Var(NoRecover, NoReInit): xbFirst(True),
        GxServer(0), GxTradeStore(0), GxConfirms(0), GxConfirm(0),

        Count(0),
        ConfirmQty(0), ConfirmPrice(0)

If xbFirst Then

        GxTradeStore = oGom.GetObject(GxServer, 3)
        GxConfirms = oGom.GetObject(GxTradeStore, 5)
        Count = oGom.GetObjectVal(GxConfirms, 3)

        GxConfirm = oGom.GetObjectByIndex(GxConfirms, 0, Count)
        ConfirmQty = oGom.GetObjectVal(GxConfirm, 4)
        ConfirmPrice = oGom.GetObjectVal(GxConfirm, 5)

        xbFirst = False

End If
```

위의 구문을 통해 ConfirmQty 와 ConfirmPrice 에 확인 수량과 확인 가격을 가져올 수 있습니다.

< GxSymbolStore >

1) 종목정보 불러오기 – KOSPI200 지수선물

```
//KOSPI200 지수선물 종목정보 불러오기
Object(NoReAlloc): oGom(DLL("STage_Root", "GosuGOM.dll", ""))                [1]
Var(NoRecover, NoReInit): xbFirst(True),                                     [2]
        GxServer(0), GxSymbolStore(0), Futures(0), GxSymbol(0),
        Code(""), KorDesc(""), gClose(0)

If xbFirst Then

        GxSymbolStore = oGom.GetObject(GxServer, 4)                          [3]
        Futures = oGom.GetObject(GxSymbolStore, 6)

        GxSymbol = oGom.GetObjectByIndex(Futures, 0, 1)

        Code = oGom.GetObjectStr(GxSymbol, 1)
        KorDesc = oGom.GetObjectStr(GxSymbol, 3)
        gClose = oGom.GetObjectVal(GxSymbol, 31)

        xbFirst = False

End If
```

[1], [2] 는 GOM 연결과 변수 선언에 대한 부분으로 앞서 예제와 동일합니다. 종목정보를 불러오기 위해서는 GxSymbolStore 에 접근해야 합니다.

GxSymbolStore = oGom.GetObject(GxServer, 4)

GxServer 의 Field ID 4 번을 통해 GxSymbolStore 에 접근할 수 있습니다.

Futures = oGom.GetObject(GxSymbolStore, 6)

이번 예제에서는 KOSPI200 지수선물 객체에 접근하고자 합니다. 위와 같이 GxSymbolStore 의 Field ID 6 번을 통해 접근이 가능합니다.

위의 Futures 는 KOSPI200 지수선물 객체를 갖고 있습니다. Collection Object 로 하위에 GxSymbol 객체를 갖고 있습니다. 하위객체는 KOSPI200 지수선물 월물별로 나뉘어 있습니다. 아래와 같이 GetObjectByIndex 를 통해 접근할 수 있습니다.

GxSymbol = oGom.GetObjectByIndex(Futures, 0, 1)

Index = 1	2	3	4
최근월물	차근월물	차차근월물	차차차근월물

Code = oGom.GetObjectStr(GxSymbol, 1)
KorDesc = oGom.GetObjectStr(GxSymbol, 3)
gClose = oGom.GetObjectVal(GxSymbol, 31)

GxSymbol 객체의 Field ID 1 번, 3 번, 31 번의 종목코드, 한글 종목명, 현재가를 Code, KorDesc, gClose 에 저장합니다.

2) 종목정보 불러오기 – 주식종목

```
//주식종목 종목정보 불러오기
Object(NoReAlloc): oGom(DLL("STage_Root", "GosuGOM.dll", ""))
Var(NoRecover, NoReInit): xbFirst(True),
        GxServer(0), GxSymbolStore(0), Stocks(0), GxSymbol(0),
        Code(""), KorDesc(""), gClose(0)

If xbFirst Then

        GxSymbolStore = oGom.GetObject(GxServer, 4)
        Stocks = oGom.GetObject(GxSymbolStore, 5)

        GxSymbol = oGom.GetObjectByIndex(Stocks, 0, 1)

        Code = oGom.GetObjectStr(GxSymbol, 1)
        KorDesc = oGom.GetObjectStr(GxSymbol, 3)
        gClose = oGom.GetObjectVal(GxSymbol, 31)

        xbFirst = False

End If
```

앞의 KOSPI200 선물 종목정보 불러오기와 유사합니다. GOM 에서는 시가총액 상위 주식종목에 대해 정보를 제공합니다. 총 25 개의 종목이 있어 Index 1 ~ 25 로 접근이 가능합니다.

```
        Stocks = oGom.GetObject(GxSymbolStore, 5)
        GxSymbol = oGom.GetObjectByIndex(Stocks, 0, 1)
```

3) 종목정보 불러오기 – KOSPI200 지수선물 스프레드

```
//KOSPI200 지수선물 스프레드 종목정보 불러오기
Object(NoReAlloc): oGom(DLL("STage_Root", "GosuGOM.dll", ""))
Var(NoRecover, NoReInit): xbFirst(True),
        GxServer(0), GxSymbolStore(0), Spreads(0), GxSymbol(0),
        Code(""), KorDesc(""), gClose(0)

If xbFirst Then

        GxSymbolStore = oGom.GetObject(GxServer, 4)
        Spreads = oGom.GetObject(GxSymbolStore, 7)

        GxSymbol = oGom.GetObjectByIndex(Spreads, 0, 1)

        Code = oGom.GetObjectStr(GxSymbol, 1)
        KorDesc = oGom.GetObjectStr(GxSymbol, 3)
        gClose = oGom.GetObjectVal(GxSymbol, 31)

        xbFirst = False

End If
```

```
        Spreads = oGom.GetObject(GxSymbolStore, 7)
        GxSymbol = oGom.GetObjectByIndex(Spreads, 0, 1)
```

GxSymbolStore 의 Field ID 7 번을 통해 KOSPI200 지수선물 스프레드에 접근할 수 있습니다.

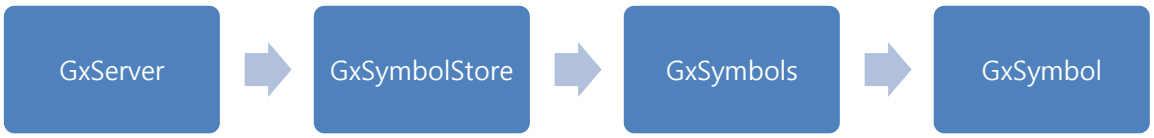
Index = 1	2	3
최근월물-차근월물	최근월물-차차근월물	최근월물-차차차근월물

4) 종목정보 불러오기 – KOSPI200 지수옵션

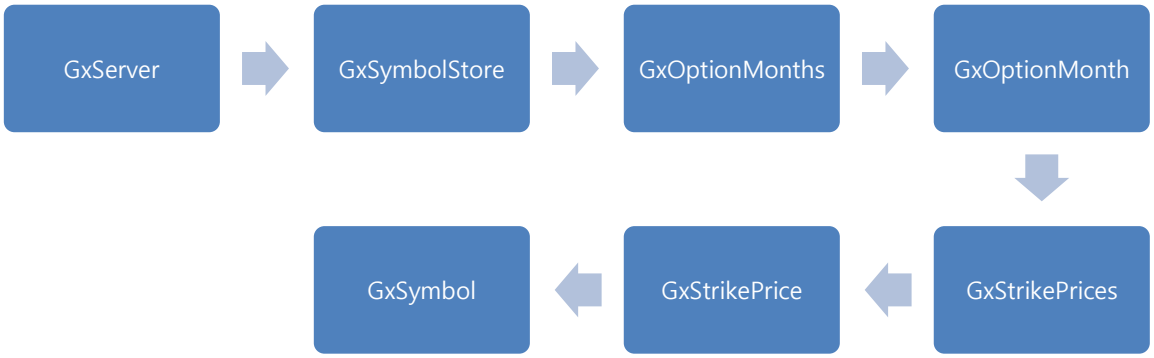
```
//KOSPI200 지수옵션 종목정보 불러오기
Object(NoReAlloc): oGom(DLL("STage_Root", "GosuGOM.dll", ""))
Var(NoRecover, NoReInit): xbFirst(True),
        GxServer(0), GxSymbolStore(0), GxOptionMonths(0), GxOptionMonth(0),
        Count1(0), Count2(0), Code(""), Desc(""), GxStrikePrices(0),
        GxStrikePrice(0), CodeCall(""), KorDescCall(""), CodePut(""), KorDescPut("")

If xbFirst Then
        GxSymbolStore = oGom.GetObject(GxServer, 4)                    [1]

        GxOptionMonths = oGom.GetObject(GxSymbolStore, 8)              [2]
        Count1 = oGom.GetObjectVal(GxOptionMonths, 3)

        GxOptionMonth = oGom.GetObjectByIndex(GxOptionMonths, 0, 1)    [3]
        Code = oGom.GetObjectStr(GxOptionMonth, 1)
        Desc = oGom.GetObjectStr(GxOptionMonth, 2)

        GxStrikePrices = oGom.GetObject(GxOptionMonth, 3)              [4]
        Count2 = oGom.GetObjectVal(GxStrikePrices, 3)

        GxStrikePrice = oGom.GetObjectByIndex(GxStrikePrices, 0, 1)    [5]
        StrikePrice = oGom.GetObjectVal(GxStrikePrice, 1)

        GxCall = oGom.GetObject(GxStrikePrice, 3)                      [6]
        GxPut = oGom.GetObject(GxStrikePrice, 4)

        CodeCall = oGom.GetObjectStr(GxCall, 3)
        CodePut = oGom.GetObjectStr(GxPut, 3)
        KorDescCall = oGom.GetObjectStr(GxCall, 4)
        KorDescPut = oGom.GetObjectStr(GxPut, 4)

        xbFirst = False
End If
```

KOSPI200 지수옵션의 경우 월물, 행사가별로 종목 수가 많습니다. 옵션객체에 대한 접근 방법과 단계는 선물객체와 약간의 차이가 있습니다.

선물객체의 접근은 아래와 같습니다. 최종 GxSymbol 객체에 도달하기 위해 아래의 단계로 접근합니다.



반면 옵션객체의 접근은 차이가 있습니다.



[1] GxSymbolStore = oGom.GetObject(GxServer, 4)

GxServer 로부터 GxSymbolStore 에 접근합니다.

[2] GxOptionMonths = oGom.GetObject(GxSymbolStore, 8)

GxSymbolStore 로부터 GxOptionMonths 에 접근합니다.

GxOptionMonths

옵션의 월물목록을 관리하는 객체입니다. 최근월물, 차근월물, 차차근월물, 차차차근월물 총 4 개의 월물을 관리합니다. GetObjectByIndex 를 통해 해당 월물목록인 GxOptiomMonth 에 접근할 수 있습니다.

[3] GxOptionMonth = oGom.GetObjectByIndex(GxOptionMonths, 0, 1)

GxOptionMonth 는 해당 월물목록객체입니다. GetObjectByIndex 를 통해 접근할 때 Index 는 아래와 같습니다.

Index = 1	2	3	4
최근월물	차근월물	차차근월물	차차차근월물

[4] GxStrikePrices = oGom.GetObject(GxOptionMonth, 3)

옵션월물목록 객체에 접근한 다음에는 행사가별로 객체에 접근할 수 있습니다. GxStrikePrices 객체는 행사가별 목록을 관리하는 객체입니다. GxOptionMonth 의 Field ID 3 번을 통해 객체에 접근할 수 있습니다.

[5] GxStrikePrice = oGom.GetObjectByIndex(GxStrikePrices, 0, 1)

GxStrikePrices 행사가별 목록 객체에서 특정 행사가의 옵션객체로 접근할 수 있습니다. 이때 GetObjectByIndex 를 통해 접근합니다. Index 는 1 부터 시작하며 1 은 현재 상장된 옵션의 행사가 중 가장 낮은 값을 가리킵니다.

[6] GxCall = oGom.GetObject(GxStrikePrice, 3)
GxPut = oGom.GetObject(GxStrikePrice, 4)

GxStrikePrice 에 접근한 후 최종적으로 콜옵션, 풋옵션에 접근할 수 있습니다. 위의 구문을 통해 가능하며 최종 콜옵션, 풋옵션객체는 GxSymbol 객체입니다. 여기서는 변수 GxCall 로 명명하였습니다.

지금까지 살펴보셨듯이 옵션객체는 **월물목록관리객체 -> 해당월물객체 -> 행사가목록관리객체 -> 해당행사가객체 -> 콜 또는 풋옵션 객체** 의 순서로 이루어집니다. 최종적으로 콜옵션, 풋옵션 객체를 획득하고 나면 GxSymbol 구조를 참고하여 정보 및 GxQuote, GxLastTick 에 접근할 수 있습니다.

다음은 KOSPI200 지수옵션 종목정보에 접근하여 ATM 옵션, ATM + 1 옵션, ATM + 2 옵션의 방식으로 객체에 접근하는 예제입니다.

```
//ATM 옵션, ATM+1 옵션, ATM+2 옵션 방식으로 옵션객체 접근하기
Object(NoReAlloc): oGom(DLL("STage_Root", "GosuGOM.dll", ""))
Var(NoRecover, NoReInit): xbFirst(True),
        GxServer(0), GxSymbolStore(0), GxOptionMonths(0), GxOptionMonth(0),
        GxK200Index(0), gOpen(0), NumOfStrike(0), GxStrikePrices(0),
        GxStrikePriceBegin(0), GxStrikePriceEnd(0), i(0), j(0),
        StrikePriceBegin(0), StrikePriceEnd(0), Index_ATM(0), StrikePrice_ATM(0),
        ATM(0), Call_ATM(0), Put_ATM(0), Call_ATM_1(0), Put_ATM_1(0),
        Call_ATM_2(0), Put_ATM_2(0), Call_ATM_3(0), Put_ATM_3(0)


If xbFirst Then

        GxSymbolStore = oGom.GetObject(GxServer, 4)
        GxOptionMonths = oGom.GetObject(GxSymbolStore, 8)
        GxOptionMonth = oGom.GetObjectByIndex(GxOptionMonths, 0, 1)
        GxK200Index = oGom.GetObject(GxSymbolStore, 13)

        gOpen = oGom.GetObjectVal(GxK200Index, 28)

        GxStrikePrices = oGom.GetObject(GxOptionMonth, 3)
        NumOfStrike = oGom.GetObjectVal(GxStrikePrices, 3)

        GxStrikePriceBegin = oGom.GetObjectByIndex(GxStrikePrices, 0, 1)
        GxStrikePriceEnd = oGom.GetObjectByIndex(GxStrikePrices, 0, NumOfStrike)

        StrikePriceBegin = oGom.GetObjectVal(GxStrikePriceBegin, 1)
        StrikePriceEnd = oGom.GetObjectVal(GxStrikePriceEnd, 1)

        For i = StrikePriceBegin To StrikePriceEnd step 2.5
                j = j + 1
                If ABS(i - gOpen) <= 1.25 Then
                        Index_ATM = j
                        StrikePrice_ATM = i
```

```
                End If
        End For

        ATM = oGom.GetObjectByIndex(GxStrikePrices, 0, Index_ATM)

        Call_ATM = oGom.GetObject(ATM, 3)
        Put_ATM = oGom.GetObject(ATM, 4)

        Call_ATM_1 = oGom.GetObject(oGom.GetObjectByIndex(GxStrikePrices, 0, Index_ATM + 1), 3)
        Put_ATM_1 = oGom.GetObject(oGom.GetObjectByIndex(GxStrikePrices, 0, Index_ATM - 1), 4)

        Call_ATM_2 = oGom.GetObject(oGom.GetObjectByIndex(GxStrikePrices, 0, Index_ATM + 2), 3)
        Put_ATM_2 = oGom.GetObject(oGom.GetObjectByIndex(GxStrikePrices, 0, Index_ATM - 2), 4)

        Call_ATM_3 = oGom.GetObject(oGom.GetObjectByIndex(GxStrikePrices, 0, Index_ATM + 3), 3)
        Put_ATM_3 = oGom.GetObject(oGom.GetObjectByIndex(GxStrikePrices, 0, Index_ATM - 3), 4)

        xbFirst = False

End If
```

GxK200Index = oGom.GetObject(GxSymbolStore, 13)
gOpen = oGom.GetObjectVal(GxK200Index, 28)

GxK200Index 객체에 접근하여 기초자산인 KOSPI200 지수의 당일 시가를 불러온 후, 이를 행사가와 비교하여 최종적으로 Call_ATM, Put_ATM 객체에 접근합니다.

ATM, ATM + 1, ATM + 2 방식으로 옵션 주문내기

앞서 ATM, ATM + 1, ATM + 2 방식으로 옵션객체에 접근한 예제와 주문메소드 실행을 이용하여 ATM, ATM + 1, ATM + 2 방식으로 옵션 주문내기가 가능합니다.

```
Object(NoReAlloc): oGom(DLL("STage_Root", "GosuGOM.dll", ""))
Var(NoRecover, NoReInit): xbFirst(True), xbOrder(True),
        GxServer(0), GxTradeStore(0), GxOrderHandler(0),
        GxSymbolStore(0), GxOptionMonths(0), GxOptionMonth(0),
        GxK200Index(0), gOpen(0), NumOfStrike(0), GxStrikePrices(0),
        GxStrikePriceBegin(0), GxStrikePriceEnd(0), i(0), j(0),
        StrikePriceBegin(0), StrikePriceEnd(0), Index_ATM(0), StrikePrice_ATM(0),
        ATM(0), Call_ATM(0), Put_ATM(0), Call_ATM_1(0), Put_ATM_1(0),
        Call_ATM_2(0), Put_ATM_2(0), Call_ATM_3(0), Put_ATM_3(0)


If xbFirst Then

        GxTradeStore = oGom.GetObject(GxServer, 3)
        GxOrderHandler = oGom.GetObject(GxTradeStore, 6)

        GxSymbolStore = oGom.GetObject(GxServer, 4)
        GxOptionMonths = oGom.GetObject(GxSymbolStore, 8)
        GxOptionMonth = oGom.GetObjectByIndex(GxOptionMonths, 0, 1)
        GxK200Index = oGom.GetObject(GxSymbolStore, 13)

        gOpen = oGom.GetObjectVal(GxK200Index, 28)

        GxStrikePrices = oGom.GetObject(GxOptionMonth, 3)
        NumOfStrike = oGom.GetObjectVal(GxStrikePrices, 3)

        GxStrikePriceBegin = oGom.GetObjectByIndex(GxStrikePrices, 0, 1)
        GxStrikePriceEnd = oGom.GetObjectByIndex(GxStrikePrices, 0, NumOfStrike)

        StrikePriceBegin = oGom.GetObjectVal(GxStrikePriceBegin, 1)
        StrikePriceEnd = oGom.GetObjectVal(GxStrikePriceEnd, 1)
```

```
        for i = StrikePriceBegin To StrikePriceEnd step 2.5
                j = j + 1
                If ABS(i - gOpen) <= 1.25 Then
                        Index_ATM = j
                        StrikePrice_ATM = i
                End If
        End For

        ATM = oGom.GetObjectByIndex(GxStrikePrices, 0, Index_ATM)

        Call_ATM = oGom.GetObject(ATM, 3)
        Put_ATM = oGom.GetObject(ATM, 4)

        Call_ATM_1 = oGom.GetObject(oGom.GetObjectByIndex(GxStrikePrices, 0, Index_ATM + 1), 3)
        Put_ATM_1 = oGom.GetObject(oGom.GetObjectByIndex(GxStrikePrices, 0, Index_ATM - 1), 4)

        Call_ATM_2 = oGom.GetObject(oGom.GetObjectByIndex(GxStrikePrices, 0, Index_ATM + 2), 3)
        Put_ATM_2 = oGom.GetObject(oGom.GetObjectByIndex(GxStrikePrices, 0, Index_ATM - 2), 4)

        Call_ATM_3 = oGom.GetObject(oGom.GetObjectByIndex(GxStrikePrices, 0, Index_ATM + 3), 3)
        Put_ATM_3 = oGom.GetObject(oGom.GetObjectByIndex(GxStrikePrices, 0, Index_ATM - 3), 4)

        xbFirst = False

End If

If xbOrder Then

        Code = oGom.GetObjectStr(Call_ATM, 1)

        oGom.AddParamStr("99999000001")
        oGom.AddParamStr(Code)
        oGom.AddParamVal(0)
        oGom.AddParamVal(1)
        oGom.AddParamVal(1)
        oGom.AddParamVal(30)

        vRtn = oGom.RunObjectVal(GxOrderHandler, 4)
```

```
        oGom.RunObject(GxOrderHandler,7)

        xbOrder = False

End If
```

굵은 글씨로 된 부분이 주문을 내는 부분입니다.

```
        Code = oGom.GetObjectStr(Call_ATM, 1)
```

Code 에 Call_ATM 의 종목코드를 저장합니다. 이후 oGom.AddParamStr 를 통해 GxOrderHandler 의 PutNewOrder 메소드에 전달합니다.

5) 종목정보 불러오기 – 국채선물

```
//국채선물 종목정보 불러오기
Object(NoReAlloc): oGom(DLL("STage_Root", "GosuGOM.dll", ""))
Var(NoRecover, NoReInit): xbFirst(True),
        GxServer(0), GxSymbolStore(0), BondFutures(0), GxSymbol(0),
        Code(""), KorDesc(""), gClose(0)

If xbFirst Then

        GxSymbolStore = oGom.GetObject(GxServer, 4)
        BondFutures = oGom.GetObject(GxSymbolStore, 10)

        GxSymbol = oGom.GetObjectByIndex(BondFutures, 0, 1)

        Code = oGom.GetObjectStr(GxSymbol, 1)
        KorDesc = oGom.GetObjectStr(GxSymbol, 3)
        gClose = oGom.GetObjectVal(GxSymbol, 31)

        xbFirst = False

End If
```

```
        GxSymbolStore = oGom.GetObject(GxServer, 4)
        BondFutures = oGom.GetObject(GxSymbolStore, 10)
```

GxSymbolStore 의 Field ID 10 번을 통해 국채선물에 접근할 수 있습니다.

Index = 1	2	3
신 3 년 국고채 최근월물	신 3 년 국고채 차근월물	신 5 년 국고채 최근월물
4	5	6
신 5 년 국고채 차근월물	신 10 년 국고채 최근월물	신 10 년 국고채 차근월물

6) 종목정보 불러오기 – 통화선물

```
//통화선물 종목정보 불러오기
Object(NoReAlloc): oGom(DLL("STage_Root", "GosuGOM.dll", ""))
Var(NoRecover, NoReInit): xbFirst(True),
        GxServer(0), GxSymbolStore(0), CurrencyFutures(0), GxSymbol(0),
        Code(""), KorDesc(""), gClose(0)

If xbFirst Then

        GxSymbolStore = oGom.GetObject(GxServer, 4)
        CurrencyFutures = oGom.GetObject(GxSymbolStore, 11)

        GxSymbol = oGom.GetObjectByIndex(CurrencyFutures, 0, 1)

        Code = oGom.GetObjectStr(GxSymbol, 1)
        KorDesc = oGom.GetObjectStr(GxSymbol, 3)
        gClose = oGom.GetObjectVal(GxSymbol, 31)

        xbFirst = False

End If
```

```
        CurrencyFutures = oGom.GetObject(GxSymbolStore, 11)
        GxSymbol = oGom.GetObjectByIndex(CurrencyFutures, 0, 1)
```

GxSymbolStore 의 Field ID 11 번을 통해 통화선물에 접근할 수 있습니다.

Index = 1 ~ 8	미국달러 선물 최근월물부터 순서대로
9 ~ 16	엔 선물 최근월물부터 순서대로
17 ~ 24	유로 선물 최근월물부터 순서대로

7) 종목정보 불러오기 – 상품선물

```
//상품선물 종목정보 불러오기
Object(NoReAlloc): oGom(DLL("STage_Root", "GosuGOM.dll", ""))
Var(NoRecover, NoReInit): xbFirst(True),
        GxServer(0), GxSymbolStore(0), ProductFutures(0), GxSymbol(0),
        Code(""), KorDesc(""), gClose(0)

If xbFirst Then

        GxSymbolStore = oGom.GetObject(GxServer, 4)
        ProductFutures = oGom.GetObject(GxSymbolStore, 12)

        GxSymbol = oGom.GetObjectByIndex(ProductFutures, 0, 1)

        Code = oGom.GetObjectStr(GxSymbol, 1)
        KorDesc = oGom.GetObjectStr(GxSymbol, 3)
        gClose = oGom.GetObjectVal(GxSymbol, 31)

        xbFirst = False

End If
```

ProductFutures = oGom.GetObject(GxSymbolStore, 12)
GxSymbol = oGom.GetObjectByIndex(ProductFutures, 0, 1)

GxSymbolStore 의 Field ID 12 번을 통해 상품선물에 접근할 수 있습니다.

Index = 1 ~ 7	금 선물 최근월물부터 순서대로
8 ~ 13	돈육 선물 최근월물부터 순서대로
14 ~ 20	미니 금 선물 최근월물부터 순서대로

8) 종목정보 불러오기 – KOSPI200 지수

```
//KOSPI200 지수 정보 불러오기
Object(NoReAlloc): oGom(DLL("STage_Root", "GosuGOM.dll", ""))
Var(NoRecover, NoReInit): xbFirst(True),
        GxServer(0), GxSymbolStore(0), K200Index(0),
        Code(""), KorDesc(""), gClose(0)

If xbFirst Then

        GxSymbolStore = oGom.GetObject(GxServer, 4)
        K200Index = oGom.GetObject(GxSymbolStore, 13)

        Code = oGom.GetObjectStr(K200Index, 1)
        KorDesc = oGom.GetObjectStr(K200Index, 3)
        gClose = oGom.GetObjectVal(K200Index, 31)

        xbFirst = False

End If
```

K200Index = oGom.GetObject(GxSymbolStore, 13)

KOSPI200 지수 정보를 불러옵니다. GxSymbolStore 의 Field ID 13 번을 통해 KOSPI200 지수에 접근할 수 있습니다. 지수선물이 아닌 현물지수이며, 여기서 K200Index 는 GxSymbol 객체입니다.

9) 종목정보 불러오기 – 선물 최근월물

```
//선물 최근월물 정보 불러오기
Object(NoReAlloc): oGom(DLL("STage_Root", "GosuGOM.dll", ""))
Var(NoRecover, NoReInit): xbFirst(True),
        GxServer(0), GxSymbolStore(0), NearestFuture(0),
        Code(""), KorDesc(""), gClose(0)

If xbFirst Then

        GxSymbolStore = oGom.GetObject(GxServer, 4)
        NearestFuture = oGom.GetObject(GxSymbolStore, 14)

        Code = oGom.GetObjectStr(NearestFuture, 1)
        KorDesc = oGom.GetObjectStr(NearestFuture, 3)
        gClose = oGom.GetObjectVal(NearestFuture, 31)

        xbFirst = False

End If
```

```
        NearestFuture = oGom.GetObject(GxSymbolStore, 14)
```

KOSPI200 지수선물 최근월물 정보를 불러옵니다. GxSymbolStore 의 Field ID 14 번을 통해 접근할 수 있습니다.

10) 종목정보 불러오기 – 옵션 최근월

```
//옵션 최근월물 종목정보 불러오기
Object(NoReAlloc): oGom(DLL("STage_Root", "GosuGOM.dll", ""))
Var(NoRecover, NoReInit): xbFirst(True),
        GxServer(0), GxSymbolStore(0), GxNearestOptMonths(0), GxNearestOptMonth(0),
        Count1(0), Count2(0), Code(""), Desc(""), GxStrikePrices(0),
        GxStrikePrice(0), CodeCall(""), KorDescCall(""), CodePut(""), KorDescPut("")

If xbFirst Then
        GxSymbolStore = oGom.GetObject(GxServer, 4)
        GxNearestOptMonth = oGom.GetObject(GxSymbolStore, 15)

        Code = oGom.GetObjectStr(GxNearestOptMonth, 1)
        Desc = oGom.GetObjectStr(GxNearestOptMonth, 2)

        GxStrikePrices = oGom.GetObject(GxNearestOptMonth, 3)
        Count2 = oGom.GetObjectVal(GxStrikePrices, 3)

        GxStrikePrice = oGom.GetObjectByIndex(GxStrikePrices, 0, 1)
        StrikePrice = oGom.GetObjectVal(GxStrikePrice, 1)

        GxCall = oGom.GetObject(GxStrikePrice, 3)
        GxPut = oGom.GetObject(GxStrikePrice, 4)

        CodeCall = oGom.GetObjectStr(GxCall, 3)
        CodePut = oGom.GetObjectStr(GxPut, 3)
        KorDescCall = oGom.GetObjectStr(GxCall, 4)
        KorDescPut = oGom.GetObjectStr(GxPut, 4)

        xbFirst = False
End If
```

GxNearestOptMonth = oGom.GetObject(GxSymbolStore, 15)

옵션최근월물객체에 접근하여 하위객체 및 정보에 접근합니다.

11) 호가정보 불러오기

```
//호가정보 불러오기
Object(NoReAlloc): oGom(DLL("STage_Root", "GosuGOM.dll", ""))
Var(NoRecover, NoReInit): xbFirst(True),
        GxServer(0), GxSymbolStore(0), NearestFuture(0), GxQuote(0),
        Code(""), KorDesc(""), gClose(0)

If xbFirst Then

        GxSymbolStore = oGom.GetObject(GxServer, 4)
        NearestFuture = oGom.GetObject(GxSymbolStore, 14)

        Code = oGom.GetObjectStr(NearestFuture, 1)
        KorDesc = oGom.GetObjectStr(NearestFuture, 3)
        gClose = oGom.GetObjectVal(NearestFuture, 31)

        GxQuote = oGom.GetObject(NearestFuture, 36)

        SellPrice1 = oGom.GetObjectValEx(GxQuote, 0, 2) / 100
        SellPrice2 = oGom.GetObjectValEx(GxQuote, 1, 2) / 100
        SellPrice3 = oGom.GetObjectValEx(GxQuote, 2, 2) / 100
        SellPrice4 = oGom.GetObjectValEx(GxQuote, 3, 2) / 100
        SellPrice5 = oGom.GetObjectValEx(GxQuote, 4, 2) / 100

        BuyPrice1 = oGom.GetObjectValEx(GxQuote, 5, 2) / 100
        BuyPrice2 = oGom.GetObjectValEx(GxQuote, 6, 2) / 100
        BuyPrice3 = oGom.GetObjectValEx(GxQuote, 7, 2) / 100
        BuyPrice4 = oGom.GetObjectValEx(GxQuote, 8, 2) / 100
        BuyPrice5 = oGom.GetObjectValEx(GxQuote, 9, 2) / 100

        xbFirst = False

End If
```

GxSymbol 에서 Field ID 36 번을 통해 GxQuote 객체에 접근할 수 있습니다. GxQuote 객체는 호가정보를 갖고 있습니다. 자세한 내용은 **4. GOM 구조도, 메소드, 이벤트**를 참고하시기 바랍니다. GxQuote 객체의 데이터에 접근하기 위해서는 GetObjectValEx 메소드를 사용해야 합니다.

```
GxQuote = oGom.GetObject(NearestFuture, 36)

SellPrice1 = oGom.GetObjectValEx(GxQuote, 0, 2) / 100
SellPrice2 = oGom.GetObjectValEx(GxQuote, 1, 2) / 100
SellPrice3 = oGom.GetObjectValEx(GxQuote, 2, 2) / 100
SellPrice4 = oGom.GetObjectValEx(GxQuote, 3, 2) / 100
SellPrice5 = oGom.GetObjectValEx(GxQuote, 4, 2) / 100

BuyPrice1 = oGom.GetObjectValEx(GxQuote, 5, 2) / 100
BuyPrice2 = oGom.GetObjectValEx(GxQuote, 6, 2) / 100
BuyPrice3 = oGom.GetObjectValEx(GxQuote, 7, 2) / 100
BuyPrice4 = oGom.GetObjectValEx(GxQuote, 8, 2) / 100
BuyPrice5 = oGom.GetObjectValEx(GxQuote, 9, 2) / 100
```

위와 같이 GxQuote 에 접근한 후 매도호가, 매수호가 정보에 접근할 수 있습니다. 다음에 이어지는 예제는 호가잔량을 불러오는 예제입니다.

```
GxQuote = oGom.GetObject(NearestFuture, 36)

SellQty0 = oGom.GetObjectValEx(GxQuote, 0, 3)
SellQty1 = oGom.GetObjectValEx(GxQuote, 1, 3)
SellQty2 = oGom.GetObjectValEx(GxQuote, 2, 3)
SellQty3 = oGom.GetObjectValEx(GxQuote, 3, 3)
SellQty4 = oGom.GetObjectValEx(GxQuote, 4, 3)
SellQty5 = oGom.GetObjectValEx(GxQuote, 5, 3)

BuyQty0 = oGom.GetObjectValEx(GxQuote, 6, 3)
BuyQty1 = oGom.GetObjectValEx(GxQuote, 7, 3)
BuyQty2 = oGom.GetObjectValEx(GxQuote, 8, 3)
BuyQty3 = oGom.GetObjectValEx(GxQuote, 9, 3)
BuyQty4 = oGom.GetObjectValEx(GxQuote, 10, 3)
BuyQty5 = oGom.GetObjectValEx(GxQuote, 11, 3)
```

```
//호가잔량 불러오기
Object(NoReAlloc): oGom(DLL("STage_Root", "GosuGOM.dll", ""))
Var(NoRecover, NoReInit): xbFirst(True),
        GxServer(0), GxSymbolStore(0), NearestFuture(0), GxQuote(0),
        Code(""), KorDesc(""), gClose(0)

If xbFirst Then

        GxSymbolStore = oGom.GetObject(GxServer, 4)
        NearestFuture = oGom.GetObject(GxSymbolStore, 14)

        Code = oGom.GetObjectStr(NearestFuture, 1)
        KorDesc = oGom.GetObjectStr(NearestFuture, 3)
        gClose = oGom.GetObjectVal(NearestFuture, 31)

        GxQuote = oGom.GetObject(NearestFuture, 36)

        SellQty0 = oGom.GetObjectValEx(GxQuote, 0, 3)
        SellQty1 = oGom.GetObjectValEx(GxQuote, 1, 3)
        SellQty2 = oGom.GetObjectValEx(GxQuote, 2, 3)
        SellQty3 = oGom.GetObjectValEx(GxQuote, 3, 3)
        SellQty4 = oGom.GetObjectValEx(GxQuote, 4, 3)
        SellQty5 = oGom.GetObjectValEx(GxQuote, 5, 3)

        BuyQty0 = oGom.GetObjectValEx(GxQuote, 6, 3)
        BuyQty1 = oGom.GetObjectValEx(GxQuote, 7, 3)
        BuyQty2 = oGom.GetObjectValEx(GxQuote, 8, 3)
        BuyQty3 = oGom.GetObjectValEx(GxQuote, 9, 3)
        BuyQty4 = oGom.GetObjectValEx(GxQuote, 10, 3)
        BuyQty5 = oGom.GetObjectValEx(GxQuote, 11, 3)

        xbFirst = False

End If
```

12) 틱 정보 불러오기

```
//틱 정보 불러오기
Object(NoReAlloc): oGom(DLL("STage_Root", "GosuGOM.dll", ""))
Var(NoRecover, NoReInit): xbFirst(True),
        GxServer(0), GxSymbolStore(0), NearestFuture(0), GxLastTick(0),
        Code(""), KorDesc(""), gClose(0)

If xbFirst Then

        GxSymbolStore = oGom.GetObject(GxServer, 4)
        NearestFuture = oGom.GetObject(GxSymbolStore, 14)

        Code = oGom.GetObjectStr(NearestFuture, 1)
        KorDesc = oGom.GetObjectStr(NearestFuture, 3)
        gClose = oGom.GetObjectVal(NearestFuture, 31)

        GxLastTick = oGom.GetObject(NearestFuture, 37)

        gPrice = oGom.GetObjectVal(GxLastTick, 3)
        gVolume = oGom.GetObjectVal(GxLastTick, 4)

        xbFirst = False

End If
```

GxSymbol 에서 Field ID 37 번을 통해 GxLastTick 객체에 접근할 수 있습니다. GxLastTick 객체는 틱 데이터 정보를 갖고 있습니다. GxLastTick 의 Field ID 3 번과 4 번은 각각 현재가와 체결량으로 틱가격, 틱체결량이 됩니다.

```
        GxLastTick = oGom.GetObject(NearestFuture, 37)
        gPrice = oGom.GetObjectVal(GxLastTick, 3)
        gVolume = oGom.GetObjectVal(GxLastTick, 4)
```