Contents

VertexFX Bridge Language

VertexFX Bridge Language (VBL) Introduction

VThe VBL Scripting Language is the soul of the VertexFX Bridge. It is a procedural programming language that is developed specifically to serve brokers in their risk management behavior according to their policies. The language provides the framework required to build sophisticated dealing and risk management scripts. VBL offers great advanced features that represent the future of risk management. VertexFX Bridge has penetrated the risk management world by its advanced scripting VB.NET-Like language, VBL.

VBL Editor is your guide to help you how to build your script and compile it using its built-in compiler to make sure your script is errors free.

Methods

AcceptOrder

Syntax

```
Public Function AcceptOrder(ByVal BOOrderID As String,
Optional ByVal AtPrice As Double =0) As TransResult
```

This function accepts a certain market order.

Parameters

Key	Description
BOOrderID	ID of the order
AtPrice	Order price

Return value

Returns an integer Value of type TransResult.

Sample

```
Public Sub BOChatOrderReceived(BOOrder As Order)

     Msgbox (BO. AcceptOrder(BOOrder.BOOrderID).Succeeded)

End Sub
```

RejectOrder

Syntax

```
Public Function RejectOrder(ByVal BOOrderID As String) As
VertexFXBridgeAPI.TransResult
```

This function rejects a certain market order.

Parameters

Key	Description
BOOrderID	ID of the order

Return value

Returns an integer Value of type TransResult.

Sample

```
Public Sub BOChatOrderReceived(BOOrder As Order)

  Msgbox (BO. RejectOrder(BOOrder.BOOrderID).Succeeded)

End Sub
```

Connect

Syntax

```
Public Sub Connect()
```

This sub connects the Backoffice.

Sample

```
 Public Sub BOConnectionState (ByVal IsConnected As Boolean)

    If Not IsConnected Then
           BO.Connect ()
    End If

 End Sub
```

Disconnect

Syntax

```
Public Sub Disconnect()
```

This sub disconnects the Backoffice.

Sample

```
Public Sub BOConnectionState (ByVal IsConnected As Boolean)

    If IsConnected Then
       BO. Disconnect ()
    End If

End Sub
```

SwitchChat

Syntax

```
Public Sub SwitchChat(ByVal State As Boolean)
```

This sub enables or disables the chat screen of the Backoffice.

Parameters

Key	Description
State	The status of the chatting screen

Sample

```
Public Sub BOChatStatus (ByVal ChatOn As Boolean)

    If ChatOn = False Then
       BO.SwitchChat (True)
    End If

End Sub
```

Symbol

Syntax

```
Public Function Symbol(ByVal Name As String) As Symbol
```

```
Public Function Symbol(ByVal Index As Integer) As Symbol
```

This function gets the symbol by its ID or returns the symbol by its index.

Parameters

Key	Description
Name	Name of the symbol
Index	Index of the symbol

Return value

Returns an object of type Symbol.

Sample

```
Public sub main  ()

    Msgbox (BO.Symbol ("Gold").Ask)

End Sub
```

SymbolsCount

Syntax

```
Public Function SymbolsCount() As Long
```

This function returns the symbols count.

Return value

Returns a number of type Long.

Sample

```
Public Sub main()

   Msgbox (BO.SymbolsCount())

End Sub
```

GetBO

Syntax

```
Public Function GetBO() As VertexFXBO
```

This function is used to return the Backoffice object.

Return value

Returns a object of type VertexFXBO.

Sample

```
Public Sub main()

   GUI.addlog(BO.GetBO.LoginUsername)

End Sub
```

GetMOSymbolWaitingSeconds

Syntax

```
Public Function GetMOSymbolWaitingSeconds(ByVal SymbolID As String) As Integer
```

his function returns the waiting seconds when the market order will be covered for certain symbol.

Parameters

Key	Description
SymbolID	ID of the symbol

Return value

Returns an integer value of type integer, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

   Msgbox (BO.GetMOSymbolWaitingSeconds ("GOLD"))

End Sub
```

GetMOSymbolPriceCondition

Syntax

```
Public Function GetMOSymbolPriceCondition(ByVal SymbolID As String) As Integer
```

This function returns the market order price condition for certain symbol.

Parameters

Key	Description
SymbolID	ID of the symbol

Return value

Returns an integer value of type integer, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

   Msgbox (BO.GetMOSymbolPriceCondition ("GOLD"))

End Sub
```

GetMOSymbolAlwaysCoverOnLiquidation

Syntax

```
Public Function GetMOSymbolAlwaysCoverOnLiquidation
(ByVal SymbolID As String) As TransResult
```

This function returns the market order state if it is always covered on liquidation for a certain symbol.

Parameters

Key	Description
SymbolID	ID of the symbol

Return value

Returns value of type TransResult.

Sample

```
Public Sub main()

   Msgbox (BO.GetMOSymbolAlwaysCoverOnLiquidation("GOLD"))

End Sub
```

GetMOBetterPipsBuy

Syntax

```
Public Function GetMOBetterPipsBuy(ByVal SymbolID As String)
As Integer
```

This function returns the better pips amount of the covered market order with buy type for a certain symbol when the "Price Condition" option is set to "Cover when LP price is better by ...".

Parameters

Key	Description
SymbolID	ID of the symbol

Return value

Returns an integer value of type integer, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

   Msgbox (BO.GetMOBetterPipsBuy("GOLD"))

End Sub
```

GetMOCoverPercent

Syntax

```
Public Function GetMOCoverPercent(ByVal SymbolID As String)
As Integer
```

This function returns the percentage amount of the covered market order for a certain symbol.

Parameters

Key	Description
SymbolID	ID of the symbol

Return value

Returns an integer value of type integer, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

   Msgbox (BO.GetMOCoverPercent("GOLD"))

End Sub
```

GetMOCoverAs

Syntax

```
Public Function GetMOCoverAs(ByVal SymbolID As String) As Integer
```

This function returns the covering type of the market order for a certain symbol.

Parameters

Key	Description
SymbolID	ID of the symbol

Return value

Returns an integer value of type integer, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

   Msgbox (BO.GetMOCoverAs("GOLD"))

End Sub
```

GetMOLOCoverOffsetBuy

Syntax

```
Public Function GetMOLOCoverOffsetBuy(ByVal SymbolID As String) As Integer
```

This function returns the slippage amount of the covered market order with Buy type for certain symbol if the "Cover order as" option is set to "Limit order with offset..".

Parameter

Key	Description
SymbolID	ID of the symbol

Return value

Returns an integer value of type integer, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

   Msgbox (BO.GetMOLOCoverOffsetBuy("GOLD"))

End Sub
```

GetMOAcceptWithPrice

Syntax

```
Public Function GetMOAcceptWithPrice(ByVal SymbolID As String) As Integer
```

This function returns the accepted price type of the market order for a certain symbol.

Parameters

Key	Description
SymbolID	ID of the symbol

Return value

Returns an integer value of type integer, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

   Msgbox (BO.GetMOAcceptWithPrice("GOLD"))

End Sub
```

GetMOAcceptAtOffsetBuy

Syntax

```
Public Function GetMOAcceptAtOffsetBuy(ByVal SymbolID As String) As Integer
```

This function returns the slippage amount of the accepted market order for a certain symbol.

Parameters

Key	Description
SymbolID	ID of the symbol

Return value

Returns an integer value of type integer, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

   Msgbox (BO.GetMOAcceptAtOffsetBuy("GOLD"))

End Sub
```

GetLOSymbolWaitingSeconds

Syntax

```
Public Function GetLOSymbolWaitingSeconds(ByVal SymbolID As String) As Integer
```

This function returns the waiting seconds for a certain symbol when limit order will be covered.

Parameters

Key	Description
SymbolID	ID of the symbol

Return value

Returns an integer value of type integer, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

   Msgbox (BO.GetLOSymbolWaitingSeconds("GOLD"))

End Sub
```

GetLOSymbolPriceCondition

Syntax

```
Public Function GetLOSymbolPriceCondition(ByVal SymbolID As String) As Integer
```

This function returns the price condition of the covered limit order for a certain symbol.

Parameters

Key	Description
SymbolID	ID of the symbol

Return value

Returns an integer value of type integer, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

   Msgbox (BO.GetLOSymbolPriceCondition("GOLD"))

End Sub
```

GetLOBetterPipsBuy

Syntax

```
Public Function GetLOBetterPipsBuy(ByVal SymbolID As String)
As Integer
```

This function returns the pips amount of the covered limit order with Buy type for a certain symbol when the "Price Condition" option is set to "Cover when LP price is better by ...".

Parameters

Key	Description
SymbolID	ID of the symbol

Return value

Returns an integer value of type integer, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

   Msgbox (BO.GetLOBetterPipsBuy ("GOLD"))

End Sub
```

GetLOCoverPercent

Syntax

```
Public Function GetLOCoverPercent(ByVal SymbolID As String)
As Integer
```

This function returns the percentage amount of the covered limit order for a certain symbol.

Parameters

Key	Description
SymbolID	ID of the symbol

Return value

Returns an integer value, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

   Msgbox (BO.GetLOCoverPercent("GOLD"))

End Sub
```

GetLOCoverAs

Syntax

```
Public Function GetLOCoverAs(ByVal SymbolID As String) As Integer
```

This function returns the covering type of the limit order for a certain symbol.

Parameters

Key	Description
SymbolID	ID of the symbol

Return value

Returns an integer value of type integer, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

   Msgbox (BO.GetLOCoverAs("GOLD"))

End Sub
```

GetLOLOCoverOffsetBuy

Syntax

```
Public Function GetLOLOCoverOffsetBuy(ByVal SymbolID As String) As Integer
```

This function returns the slippage amount of the covered limit order with Buy type for a certain symbol if the "Cover order as" option is set to "Limit order with offset..".

Parameters

Key	Description
SymbolID	ID of the symbol

Return value

Returns an integer value of type integer, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

   Msgbox (BO.GetLOLOCoverOffsetBuy("GOLD"))

End Sub
```

GetLOAcceptWithPrice

Syntax

```
Public Function GetLOAcceptWithPrice(ByVal SymbolID As String) As Integer
```

This function returns the accepted price type of the limit order for a certain symbol.

Parameters

Key	Description
SymbolID	ID of the symbol

Return value

Returns an integer value of type integer, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

   Msgbox (BO.GetLOAcceptWithPrice("GOLD"))

End Sub
```

GetLOAcceptAtOffsetBuy

Syntax

```
Public Function GetLOAcceptAtOffsetBuy(ByVal SymbolID As String) As Integer
```

This function returns the slippage amount of the accepted limit order with Buy type for a certain symbol.

Parameters

Key	Description
SymbolID	ID of the symbol

Return value

Returns an integer value of type integer, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

   Msgbox (BO.GetLOAcceptAtOffsetBuy("GOLD"))

End Sub
```

SetMOSymbolWaitingSeconds

Syntax

```
Public Function SetMOSymbolWaitingSeconds(ByVal SymbolID As String, ByVal Value As Integer) As TransResult
```

This function sets the waiting seconds when the market order will be covered for a certain symbol.

Parameters

Key	Description
SymbolID	ID of the symbol
Value	The value of property

Return value

Returns an integer value of type integer, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

   Msgbox (BO.SetMOSymbolWaitingSeconds("GOLD" ,60))

End Sub
```

SetMOSymbolPriceCondition

Syntax

```
Public Function SetMOSymbolPriceCondition(ByVal SymbolID As String, ByVal Value As WhenToCover) As TransResult
```

This function sets the market order price condition for a certain symbol.

Parameters

Key	Description
SymbolID	ID of the symbol
Value	The value of property

Return value

Returns an integer value of type integer, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

   Msgbox (BO.SetMOSymbolPriceCondition("GOLD" ,1))

End Sub
```

SetMOSymbolAlwaysCoverOnLiquidation

Syntax

```
Public Function SetMOSymbolAlwaysCoverOnLiquidation (ByVal SymbolID As String, ByVal Value As Boolean) As TransResult
```

This function sets the market order state if it is always covered on liquidation for a certain symbol.

Parameters

Key	Description
SymbolID	ID of the symbol
Value	The value of property

Return value

Returns value of type TransResult

Sample

```
Public Sub main()

   Msgbox (BO.SetMOSymbolAlwaysCoverOnLiquidation ("GOLD",False))

End Sub
```

SetMOBetterPipsBuy

Syntax

```
Public Function SetMOBetterPipsBuy(ByVal SymbolID As String, ByVal Value As Integer) As Transresult
```

This function sets the better pips amount of the covered market order with Buy type for a certain symbol when the "Price Condition" option is set to "Cover when LP price is better by ...".

Parameters

Key	Description
SymbolID	ID of the symbol
Value	The value of property

Return value

Returns an integer value of type integer, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

   Msgbox (BO. SetMOBetterPipsBuy("GOLD",4))

End Sub
```

SetMOCoverPercent

Syntax

```
Public Function SetMOCoverPercent(ByVal SymbolID As
String, ByVal Value As Integer) As TransResult
```

This function sets the percentage amount of the covered market order for a certain symbol.

Parameters

Key	Description
SymbolID	ID of the symbol
Value	The value of property

Return value

Returns an integer value of type integer, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

   Msgbox (BO. SetMOCoverPercent("GOLD",100))

End Sub
```

SetMOCoverAs

Syntax

```
Public Function SetMOCoverAs(ByVal SymbolID As String, ByVal Value As Integer) As TransResult
```

This function sets the covering type of the market order for a certain symbol.

Parameters

Key	Description
SymbolID	ID of the symbol
Value	The value of property

Return value

Returns an integer value of type integer, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

   Msgbox (BO.SetMOCoverAs("GOLD",1))

End Sub
```

SetMOLOCoverOffsetBuy

Syntax

```
Public Function SetMOLOCoverOffsetBuy(ByVal SymbolID As String, ByVal Value As Integer) As TransResult
```

This function sets the slippage amount of the covered market order with Buy type for a certain symbol if the "Cover order as" option is set to "Limit order with offset..".

Parameters

Key	Description
SymbolID	ID of the symbol
Value	The value of property

Return value

Returns an integer value of type integer, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

   Msgbox (BO.SetMOLOCoverOffsetBuy("GOLD",1))

End Sub
```

SetMOAcceptWithPrice

Syntax

```
Public Function SetMOAcceptWithPrice(ByVal SymbolID As String, ByVal Value As Integer) As TransResult
```

This function sets the accepted price type of the market order for a certain symbol.

Parameters

Key	Description
SymbolID	ID of the symbol
Value	The value of property

Return value

Returns an integer value of type integer, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

   Msgbox (BO.SetMOAcceptWithPrice("GOLD",2))

End Sub
```

SetMOAcceptAtOffsetBuy

Syntax

```
Public Function SetMOAcceptAtOffsetBuy(ByVal SymbolID As String, ByVal Value As Integer) As TransResult
```

This function sets the slippage amount of the accepted market order with Buy type for a certain symbol.

Parameters

Key	Description
SymbolID	ID of the symbol
Value	The value of property

Return value

Returns an integer value of type integer, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

   Msgbox (BO.SetMOAcceptAtOffsetBuy("GOLD",4))

End Sub
```

SetLOSymbolWaitingSeconds

Syntax

```
Public Function SetLOSymbolWaitingSeconds(ByVal SymbolID As String, ByVal Value As Integer) As TransResult
```

This function sets the waiting seconds for a certain symbol when limit order will be covered.

Parameters

Key	Description
SymbolID	ID of the symbol
Value	The value of property

Return value

Returns an integer value of type integer, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

   Msgbox (BO. SetLOSymbolWaitingSeconds("GOLD",60))

End Sub
```

SetLOSymbolPriceCondition

Syntax

```
Public Function SetLOSymbolPriceCondition(ByVal SymbolID As String, ByVal Value As WhenTOCover) As TransResult
```

This function sets the price condition of the covered limit order for a certain symbol.

Parameters

Key	Description
SymbolID	ID of the symbol
Value	The value of property

Return value

Returns an integer value of type integer, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

   Msgbox (BO.SetLOSymbolPriceCondition("GOLD",1))

End Sub
```

SetLOBetterPipsBuy

Syntax

```
Public Function SetLOBetterPipsBuy(ByVal SymbolID As String,
ByVal Value As Integer) As TransResult
```

This function sets the pips amount of the covered limit order with Buy type for a certain symbol when the "Price Condition" option is set to "Cover when LP price is better by ...".

Parameters

Key	Description
SymbolID	ID of the symbol
Value	The value of property

Return value

Returns an integer value of type integer, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

   Msgbox (BO.SetLOBetterPipsBuy("GOLD",1))

End Sub
```

SetLOCoverPercent

Syntax

```
Public Function SetLOCoverPercent(ByVal SymbolID As String,
ByVal Value As Integer) As TransResult
```

This function sets the percentage amount of the covered limit order for a certain symbol.

Parameters

Key	Description
SymbolID	ID of the symbol
Value	The value of property

Return value

Returns an integer value of type integer, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

   Msgbox (BO.SetLOCoverPercent("GOLD",90))

End Sub
```

SetLOCoverAs

Syntax

```
Public Function SetLOCoverAs(ByVal SymbolID As String, ByVal Value As Integer) As TransResult
```

This function sets the covering type of the limit order for a certain symbol.

Parameters

Key	Description
SymbolID	ID of the symbol
Value	The value of property

Return value

Returns an integer value of type integer, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

   Msgbox (BO.SetLOCoverAs("GOLD",1))

End Sub
```

SetLOLOCoverOffsetBuy

Syntax

```
Public Function SetLOLOCoverOffsetBuy(ByVal SymbolID As String, ByVal Value As Integer) As TransResult
```

This function sets the slippage amount of the covered limit order for a certain symbol if the "Cover order as" option is set to "Limit order with offset..".

Parameters

Key	Description
SymbolID	ID of the symbol
Value	The value of property

Return value

Returns an integer value of type integer, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

   Msgbox (BO. SetLOLOCoverOffsetBuy("GOLD",1)

End Sub
```

SetLOAcceptWithPrice

Syntax

```
Public Function SetLOAcceptWithPrice(ByVal SymbolID As
String, ByVal Value As AcceptOrderWithPrice) As TransResult
```

This function set the accepted price type of the limit order for a certain symbol.

Parameters

Key	Description
SymbolID	ID of the symbol
Value	The value of property

Return value

Returns an integer value of type integer, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

   Msgbox (BO. SetLOAcceptWithPrice("GOLD",1))

End Sub
```

SetLOAcceptAtOffsetBuy

Syntax

```
Public Function SetLOAcceptAtOffsetBuy(ByVal SymbolID As String, ByVal Value As Integer) As TransResult
```

This function sets the slippage amount of the accepted limit order with Buy type for a certain symbol.

Parameters

Key	Description
SymbolID	ID of the symbol
Value	The value of property

Return value

Returns an integer value of type integer, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

   Msgbox (BO. SetLOAcceptAtOffsetBuy("GOLD",1))

End Sub
```

SaveSymbolSettings

Syntax

```
Public Function SaveSymbolSettings(ByVal SymbolID As String)
As TransResult
```

This function saves the settings for a certain symbol.

Parameters

Key	Description
SymbolID	ID of the symbol

Return value

Returns an integer value of type integer, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

   Msgbox (BO. SaveSymbolSettings("GOLD").Succeeded)

End Sub
```

LPNewLimitOrder

Syntax

```
Public Function LPNewLimitOrder(ByVal LPDescreption As String, ByVal LOTypeAs VertexFXBridgeAPI.LimitOrderType, ByVal Amount As Double, ByValLPSymbolName As String, ByVal  Price As Double, ByVal LPAccount As Long, ByValBOOrderID As String) As TransResult
```

This function creates a new limit order on a certain symbol on the liquidity provider side.

Parameters

Key	Description
LPDescreption	ID of the liquidity provider
LOType	Type of order of type LimitOrderType
Amount	Amount of the order
LPSymbolName	Symbol name
Price	Order open price
LPAccount	ID of liquidity provider account
BOOrderID	Backoffice order ID

Return value

Returns TransResult value.

Sample

```
Public Sub main()

   Msgbox (LP. LPNewLimitOrder ("LP",1,1, "GOLD",1500 ,4454 ,4545).Succeeded)

End Sub
```

LPNewOrder

Syntax

```
Public Function LPNewOrder(ByVal LPDescreption As
String, ByVal BuySell AsOperationType, ByVal Amount As
Double, ByVal LPSymbolName As String, ByValLPAccountID As
String, ByVal BOOrderID As String) As TransResult
```

This function creates a new order for a certain symbol on the liquidity provider side.

Parameters

Key	Description
LPDescreption	ID of the liquidity provider
BuySell	The type of the order of type OperationType
Amount	Amount of the order
LPSymbolName	Symbol name
LPAccount	ID of liquidity provider account
BOOrderID	Backoffice order ID

Return value

Returns TransResult value.

Sample

```
Public Sub main()

   Msgbox (LP. LPNewLimitOrder ("LP",1,1,"GOLD", 1500,4454 )
.Succeeded))

End Sub
```

LPConnect

Syntax

```
Public Function LPConnect(ByVal LPDescreption As String) As TransResult
```

This function connects a certain liquidity provider.

Parameters

Key	Description
LPDescreption	ID of the liquidity provider

Return value

Returns TransResult value.

Sample

```
Public Sub main()

   Msgbox (LP. Connect("LP").Succeeded)

End Sub
```

LPDisconnect

Syntax

```
Public Function LPDisconnect(ByVal LPDescreption As String)
As TransResult
```

This function disconnects a certain liquidity provider.

Parameters

Key	Description
LPDescreption	ID of the liquidity provider

Return value

Returns TransResult value.

Sample

```
Public Sub main()

   Msgbox (LP.Disconnect("LP").Succeeded)

End Sub
```

GetLP

Syntax

```
Public Function GetLP(ByVal LPDescreption As String) As VertexFXBridgeAPI.VertexFXLP
```

This function returns a certain liquidity provider object.

Parameters

Key	Description
LPDescreption	ID of the liquidity provider

Return value

Returns VertexFXBridgeAPI.VertexFXLP object, when passing an invalid LP description its returns nothing.

Sample

```
Public Sub main()

   Msgbox (LP. GetLP("LP").Description)

End Sub
```

GetLPs

Syntax

```
Public Function GetLPs(ByVal LPDescreption As String) As
Collection
```

This function returns a collection of liquidity providers objects.

Parameters

Key	Description
LPDescreption	ID of the liquidity provider

Return value

Returns collection of objects, to get VertexFXBridgeAPI.VetexFXLP object use "GetLPs.item(X).LP" .

Sample

```
Public Sub main()

   Msgbox (LP.GetLPs.item(1).LP.Description)

End Sub
```

HandleOrder

Syntax

```
Public
function HandleOrder(ByVal BOOrder As Order) As TransResult
```

This sub procedure is used to send the BackOffice order to the Bridge chatting screen.

Parameters

Key	Description
BOOrder	The Object of the Order

GetMOBetterPipsSell

Syntax

```
Public Function GetMOBetterPipsSell(ByVal SymbolID As String) As Integer
```

This function returns the better pips amount of the covered market order with sell type for certain symbol when the "Price Condition" option is set to "cover when LP price is better by…".

Parameters

Key	Description
SymbolID	ID of the symbol

Return value

Returns an integer value of type integer, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

   Msgbox (BO.GetMOBetterPipsSell("GOLD"))

End Sub
```

GetMOLOCoverOffsetSell

Syntax

```
Public Function GetMOLOCoverOffsetSell(ByVal SymbolID As String) As Integer
```

This function returns the slippage amount of the covered market order with Sell type for certain symbol if the "cover order as" option is set to "limit order with offset…"

Parameters

Key	Description
SymbolID	ID of the symbol

Return value

Returns an integer value of type integer, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

   Msgbox (BO.GetMOLOCoverOffsetSell("GOLD"))

End Sub
```

GetMOAcceptAtOffsetSell

Syntax

```
Public Function GetMOAcceptAtOffsetSell(ByVal SymbolID As String) As Integer
```

This function return the slippage amount of accepted market order of Sell type for certain Symbol.

Parameters

Key	Description
SymbolID	ID of the symbol

Return value

Returns an integer value of type integer, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

   Msgbox (BO.GetMOAcceptAtOffsetSell("GOLD"))

End Sub
```

SetMOBetterPipsSell

Syntax

```
Public Function SetMOBetterPipsSell(ByVal SymbolID As String,
ByVal value As Integer) As TransResult
```

This function sets the better pips amount of the covered market order with Sell type for certain symbol when the "Price Condition" option is set to "Cover when LP price is better by ..."

Parameters

Key	Description
SymbolID	ID of the symbol
Value	The value of property

Return value

Returns an integer value of type integer, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

   Msgbox (BO.SetMOBetterPipsSell("GOLD",4))

End Sub
```

SetMOLOCoverOffsetSell

Syntax

```
Public Function SetMOLOCoverOffsetSell(ByVal SymbolID As String, ByVal valueAs Integer) As TransResult
```

This function sets the better pips amount of the covered market order with Sell type for certain symbol when the "Price Condition" option is set to "Cover when LP price is better by ..."

Parameters

Key	Description
SymbolID	ID of the symbol
Value	The value of property

Return value

Returns an integer value of type integer, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

  Msgbox (BO.SetMOLOCoverOffsetSell("GOLD",1))

End Sub
```

SetMOAcceptAtOffsetSell

Syntax

```
Public Function SetMOAcceptAtOffsetSell(ByVal SymbolID As String, ByVal valueAs Integer) As TransResult
```

This function sets the slippage amount of the accepted market order with Sell type for certain symbol.

Parameters

Key	Description
SymbolID	ID of the symbol
Value	The value of property

Return value

Returns an integer value of type integer, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

   Msgbox (BO.SetMOAcceptAtOffsetSell("GOLD",1))

End Sub
```

GetLOBetterPipsSell

Syntax

```
Public Function GetLOBetterPipsSell(ByVal SymbolID As String)
As Integer
```

This function returns the pips amount of covered limit order with Sell type for certain symbol when the "Price Condition" option is set to "Cover when LP price is better by..."

Parameters

Key	Description
SymbolID	ID of the symbol

Return value

Returns an integer value of type integer, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

   Msgbox (BO. GetLOBetterPipsSell("GOLD"))

End Sub
```

GetLOLOCoverOffsetSell

Syntax

```
Public Function GetLOLOCoverOffsetSell(ByVal SymbolID As String) As Integer
```

This Function returns the slippage of the covered limit order with Sell type for certain symbol if the "Cover order as" option is set to "Limit order with offset..."

Parameters

Key	Description
SymbolID	ID of the symbol

Return value

Returns an integer value of type integer, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

   Msgbox (BO. GetLOLOCoverOffsetSell("GOLD"))

End Sub
```

GetLOAcceptAtOffsetSell

Syntax

```
Public Function GetLOAcceptAtOffsetSell(ByVal SymbolID As String) As Integer
```

This function returns the slippage amount of the accepted limit order with Sell type for certain symbol.

Parameters

Key	Description
SymbolID	ID of the symbol

Return value

Returns an integer value of type integer, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

   Msgbox (BO. GetLOAcceptAtOffsetSell("GOLD"))

End Sub
```

SetLOBetterPipsSell

Syntax

```
Public Function SetLOBetterPipsSell(ByVal SymbolID As
String,ByVal value As Integer) As TransResult
```

This function sets the pips amount of the covered limit order with sell type for certain symbol when the "Price Condition" option is set to "Cover when LP price is better by.

Parameters

Key	Description
SymbolID	ID of the symbol
Value	The value of property

Return value

Returns an integer value of type integer, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

   Msgbox (BO. SetLOBetterPipsSell("GOLD"))

End Sub
```

SetLOLOCoverOffsetSell

Syntax

```
Public Function SetLOLOCoverOffsetSell(ByVal SymbolID As
String,ByVal valueAs Integer) As TransResult
```

This function sets the slippage amount of the covered limit order with Sell type for certain symbol if the "cover order as" option is set to "Limit order with offset…".

Parameters

Key	Description
SymbolID	ID of the symbol
Value	The value of property

Return value

Returns an integer value of type integer, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

   Msgbox (BO.SetLOLOCoverOffsetSell("GOLD",1))

End Sub
```

SetLOAcceptAtOffsetSell

Syntax

```
Public Function SetLOAcceptAtOffsetSell(ByVal SymbolID As String, ByVal value As Integer) As TransResult
```

This function sets the slippage amount of accepted limit order with Sell type for a certain symbol.

Parameters

Key	Description
SymbolID	ID of the symbol
Value	The value of property

Return value

Returns an integer value of type integer, when passing an invalid symbol ID it returns -1.

Sample

```
Public Sub main()

   Msgbox (BO.SetLOAcceptAtOffsetSell("GOLD",1))

End Sub
```

ExecuteQuery

Syntax

```
Public Function ExecuteQuery (ByVal CommandText As
String) As System.Data.DataRowCollection
```

This method return the count of rows that selected from database.

Parameters

Key	Description
CommandText	The select command

Sample

```
Public Sub main ()

   Dim Result As Data.DataRowCollection
   Result = DB.ExecuteQuery ("SELECT * FROM History WHERE
(BOOrderStatus = 'ACCEPTED') ")
   Msgbox (Result)

End Sub
```

ExecuteNonQuery

Syntax

```
Public Function ExecuteNonQuery(ByVal CommandText As String)
```

This method use to insert into database, update felids in database, delete from database.

Parameters

Key	Description
CommandText	insert, update, delete command

Sample

```
Public Sub main ()

  Dim Query As string ="UPDATE History SET BOOrder Ticket= '"
&1234 & " WHERE (History.BOOrderID= '" & -1 &"')"
  DB.ExecuteNonQuery (Query)

  End Sub
```

AddLog

Syntax

```
Public sub AddLog(ByVal Entry As String)
```

This method use to add String to the log.

Parameters

Key	Description
Entry	The string that will add to the log

Sample

```
Public Sub SymbolSettingSave ()

  GUI.AddLog("Save groups settings...")

End Sub
```

Data Types

TransResult

This class is to indicate the transaction result, if it is succeeded or not.

Properties

Property	Description
Succeeded	Boolean whether succeeded or not
Message	The failing reason, just used when returning failing result

Symbol

This class represents the Symbol.

Properties

Property	Description
Name	The name of the symbol
Ask	Ask price
Bid	Bid price
Spread	used to return the spread value (Ask – bid), as value of type String
PipLocation	used to get the pip location value for the symbol as value of type Integer.
PipsValue	Function used to get the pip value for the given pip location as value of type Double. I.e PipsValue (5) for EUR/USD is 0.00005

Order

Properties

Part	Description
AccountID	Account ID
AccountLiquidationOrder	Boolean variable telling you that the order from the system or not, this value is True in the case of account liquidation, otherwise it is False
Amount	Lots Amount
BOOrderID	Order ID
NewOrClose	To recognize the order (New order, or Liquidate order) of type MarketOrderTypeEnum
BOSymbolName	Symbol Name
BuySell	Buy or Sell of type OperationTypeEnum
ChattingOrderType	Value of type ChattingOrderTypeEnum
LimitOrderType	Value of type LimitStopOrderType
Price	Requested price as double
OpenPrice	Open price as double, if NewOrClose is of type CloseOrder
HitPrice	The order hit price if the MarketOrLimitOrder is of type LimitOrderType, SLOrderType or TPOrderType
RefHitPrice	Reference symbol hit Price for this order if the MarketOrLimitOrder is of type LimitOrderType, SLOrderType or TPOrderType
OrderTime	The order requested time (in server time)
OpenAmount	Open amount as double, if NewOrClose is of type CloseOrder
Tag	Empty variable
VBLHandiling	You can set this Boolean to true if you want to handle this order using VBL
HandlingLPDescription	You can assign this with LP description to handling orders according to LP description settings

Account

This class represents the Account object that has the information for this account.

Properties

Part	Description
AccountID	The account ID for the account.
ClientID	The Client ID who own this account.
FirstName	The first name of the client who owns this account.
SecondName	The second name of the client who owns this account.
ThirdName	The third name of the client who owns this account.
LastName	The last name of the client who owns this account.
FullName	The full name of the client who owns this account.
DemoAccount	The account type; True, if demo. False, if real.
Credit	The credit for the account.
Balance	The balance for the account.
Equity	The Equity for the account.
MarginLevel	The margin level for the account.
EffectiveMargin	The Effective margin for account.
MarginRequirement	The margin requirement for the account.
FloatingProfitLoss	The floating profit loss of the account. Read-only

Position

Properties

Part	Description
BuySell	Define buy or sell
ClosePrice	Closing price
Commission	Commission
FloatingProfitLoss	Profit loss value
Interest	Interest Value
Liquidated	Is Liquidated Account?
Lots	Amount of lots
PosTime	Time of open position
Price	Price of open poition
RefClosePrice	Close Price of referenced symbol
SymbolID	Symbol ID
SymbolName	symbol Name
Ticket	Ticket ID Number

Client

This class represents the Client object that has the information for this client.

Properties

Part	Description
ClientID	The Client ID who own this account.
FirstName	The first name of the client who owns this account.
SecondName	The second name of the client who owns this account.
ThirdName	The third name of the client who owns this account.
LastName	The last name of the client who owns this account.
FullName	The full name of the client who owns this account.

OperationType Enum

Properties

Value	Constant	Description
-1	SellType	For sell orders
1	BuyType	For buy orders
-2	SellStop	For sell stop orders
2	BuyStop	For buy Stop orders

MarketOrderType Enum

Properties

Value	Description
1	For new orders
2	For closing orders

ChattingOrderType Enum

Properties

Value	Description
1	For market orders
2	For limit/stop orders
3	For stop loss orders
4	For take profit orders

LimitStopOrderType Enum

Properties

Value	Description
1	For BuyLimit orders
2	For BuyStop orders
3	For SellLimit orders
4	For SellStop orders

LimitOrderType Enum

Properties

Value	Description
1	For buy limit orders
-1	For sell limit orders
2	For buy stop orders
-2	For sell stop orders

TransResult

This class is to indicate the transaction result, if it is succeeded or not.

Properties

Property	Description
Succeeded	Boolean whether succeeded or not
Message	The failing reason, just used when returning failing result