Fortex 6 AlgoX User Guide(C#)

Version 1.0

Table of Contents

1. Global Functions

1.1 GetCurrentAccount

Description:

- **GetCurrentAccount ()**

Retrieve the current account object.

Parameters:

None, function takes no parameters.

Example

C#:

```
using System;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   //the base class of trade operation
   public  class  Sample: UserScript
   {
      public  Sample()
      {
         var account = Account.GetCurrentAccount();
         string name=account.GetAccountName();
         Print(name);
      }
   }
}
```

1.2 GetQuote

Description:

- **GetQuote** (*sQuoteType*)

Retrieve current bid or ask price.

Parameters：

sQuoteType: Required. A String value. The text string constants representing the price type. The following constants are available:

Constant	Comment
Bid	Bid price
Ask	Ask price
last	The last transaction price

Note：

If the return value is 0 or a negative integer value, it means the function failed to retrieve the quote.

Example

C#:

```
using System;
using  Algo.API;
using  Algo.API.Order;

namespace Test
{
   public  class  Sample: UserScript
   {
      public override void OnInit()
      {
            API.EnableTrading();
      }

      public override void OnTick(int index)
      {
       var quote=API.GetQuote("ASK");
       Print("quote:"+quote);
      }
   }
```

```
}
```

1.3 SetDefaultExpiry

Description:

- **SetDefaultExpiry (*sExpiry*)**

Set the default value of Time-In-Force (TIF) setting for the order.

Parameters:

sExpiry: Required. Expiration for the order. The following constants are available:

Constant	Comment
FOK	TIF Setting is “FOK”
GTC	TIF Setting is “GTC”
IOC	TIF Setting is “IOC”

Example

C#:

```
using System;
using Algo.API;
using Algo.API.Order;
namespace Test
{
public class Sample: UserScript
{
public Sample()
{
API.SetDefaultExpiry("IOC");
String Expiry=API.GetDefaultExpiry();
Print("DefaultExpiry:"+Expiry) ;

}
}
}
```

1.4 GetDefaultExpiry

Description:

- **GetDefaultExpiry ()**

Retrieve the EXPIRY value.

Parameters:

None, function takes no parameters.

Example

C#:

```
using System;
using Algo.API;
using Algo.API.Order;
namespace Test
{
public class Sample: UserScript
{
public Sample()
{
String Expiry=API.GetDefaultExpiry();
Print("DefaultExpiry:"+Expiry) ;
}
}
}
```

1.5 GetOldestBarIndex

Description:

GetOldestBarIndex ()

Returns the bar index of the oldest bar in the series.
This will always return a negative integer value unless no data exists for the symbol in the chart window, in which case it will return null.

Parameters:

None, function takes no parameters.

Example

C#:

```
using Algo.API;
using Algo.API.Order;
namespace Test
```

```
{
public class Sample: UserScript
{
public Sample()
{
int OldBarindex=API.GetOldestBarIndex();
Print("OldestBarIndex:"+OldBarindex);
}
}
}
```

1.6 GetCurrentBarIndex

Description:

- **GetCurrentBarIndex ()**

Returns the current offset into the price series that is loaded in the chart.

Parameters:

None, function takes no parameters.

Example

C#:

```
using System;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   public  class  Sample: UserScript
   {
      public override void OnInit()
      {

      }
      public Sample()
      {
      int CurrentBarindex=API.GetCurrentBarIndex();
      Print("CurrentBarIndex:"+ CurrentBarindex);
      }
   }
}
```

1.7 SetDefaultQuantity

Description:

- **SetDefaultQuantity (***iQuantity***)**

Set the default QUANTITY value of the order.

Parameters:

iQuantity: Required. A positive integer value.

Example

C#:

```
using System;
using Algo.API;
using Algo.API.Order;
namespace Test
{
public class Sample: UserScript
{
public Sample()
{
API.SetDefaultQuantity(10000);
int Quantity=API.GetDefaultQuantity();
Print("DefaultQuantity:"+Quantity) ;
}
}
}
```

1.8 GetDefaultQuantity

Description:

- **GetDefaultQuantity ()**

Retrieve the default QUANTITY value of the order.

Parameters:

None, function takes no parameters.

Example

C#:

```
using System;
using Algo.API;
using Algo.API.Order;
namespace Test
{
public class Sample: UserScript
{
public Sample()
{
API.SetDefaultQuantity(10000);
int Quantity=API.GetDefaultQuantity();
Print("DefaultQuantity:"+Quantity) ;
}
}
}
```

1.9 GetValue

Description:

- **GetValue (***sBarType*, *iBarIndex* [,*iNumBars*]**)**

Returns the value of the specific price of the specific bar index relative to the bar currently being processed.

Parameters:

sBarType: Required. A String value. The following constants are available:

Constant	Comment
open	The open price of specified Bar.
high	The high price of specified Bar.
low	The low price of specified Bar.
close	The close price of specified Bar.
average	The average price of specified Bar.
volume	The volume value of specified Bar.
year	The year value of specified Bar.
month	The month value of specified Bar.
day	The day value of specified Bar.
hour	The hour value of specified Bar.
minute	The minute value of specified Bar.

second	The second value of specified Bar.

iBarIndex: Required. The bar index of series to retrieve.

iNumBars: Optional. Number of bars of data to return.

Note：

GetValue (barType, 0, 0) don't do any processing, returns null values.

Example

C#：

```
using System;
using System.Collections.Generic;
using   Algo.API;
using   Algo.API.Order;
using Algo.API.Indicators;
namespace Test
{
    public   class   Sample: UserScript
    {
        public override void OnTick(int index)
        {
            var Highvalue0= MarketSeries.High.GetValue(0);
            Print("The high price of the bar is: " + Highvalue0 );
        }
    }
}
```

1.10 IsHistoryBar

Description:

- **IsHistoryBar ()**

See whether the bar is a historical data bar or not. Retrieve the BOOL value。

Parameters:

None, function takes no parameters.

Example

C#：

```
using System;
```

```
using Algo.API;
using Algo.API.Order;
namespace Test
{
public class Sample: UserScript
{
public Sample()
{
if (API.IsHistoryBar ())
{
Print ("here is history data update");
}
else
{
Print ("here is real-time update");
}
}
}
}
```

1.11 GetCurSymbol

Description:

- **GetCurSymbol ()**

Returns the name of the symbol whose quote is being updated.
Returns an empty string when no quote is being updated.

Parameters:

None, function takes no parameters.

Example

C#:

```
using System;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   public  class  Sample: UserScript
   {
      public override void OnInit()
```

```
    {
      API.EnableTrading();
    }
    public override void OnTick(int index)
    {
      var sym=API.GetCurSymbol();
      Print("sym:" + sym );
    }
  }
}
```

1.12 Sleep

Description:

- **Sleep (*millisecond*)**

Make the script sleep for a period of the time.

Parameters:

millisecond: Required. The amount of time to sleep.

Example

C#:

```
using System;
using System;
using Algo.API;
using Algo.API.Order;
namespace Test
{
public class Sample: UserScript
{
public override void OnInit()
{
API.EnableTrading();
Print("before:"+System.DateTime.Now.ToString());
API.Sleep(10000);
Print("After:"+System.DateTime.Now.ToString());
}
}
}
```

1.13 GetBidSize

Description:

- **GetBidSize ()**

Retrieve current symbol's bid size.

Parameters:

None, function takes no parameters.

Example

C#:

```
using System;
using  Algo.API;
using  Algo.API.Order;

namespace Test
{
   public  class  Sample: UserScript
   {
      public override void OnInit()
      {
        API.EnableTrading();
      }
      public override void OnTick(int index)
      {
        var bidsize=API.GetBidSize();
         Print("bidsize:"+bidsize);
      }
   }
}
```

1.14 GetAskSize

Description:

- **GetAskSize ()**

Retrieve current symbol's ask size.

Parameters:

None, function takes no parameters.

Example

C#:

```
using System;
using  Algo.API;
using  Algo.API.Order;

namespace Test
{
   public  class  Sample: UserScript
   {
      public override void OnInit()
      {
        API.EnableTrading();
      }
      public override void OnTick(int index)
      {
        var asksize=API.GetAskSize();
         Print("asksize:"+asksize);
      }
   }
}
```

1.15 GetCurTime

Description:

- **GetCurTime ()**

Returns the actual time string in "HH:MM:SS" format

Parameters:

None, function takes no parameters.

Example

C#:

```
Comment：Ues 'System.DateTime.Now.ToString()' to get the time value in C#.
```

1.16 GetQuoteInfo

Description:

- **GetQuoteInfo ()**

Returns the last quote object, return value is a Quote object.

Parameters:

None, function takes no parameters.

Example

C#:

No 'getQuoteInfo' to get quote info in C#.

1.17 GetTick

Description:

- **getTick (*iTicktype*)**

Returns the tick information for a ticktype (quotetick).
The return value is an integer value, as followed:

Return value	Comment
-2	Compared with next- to- last quote, the price is descending; but compared with last quote, the price has not changed.
-1	Compared with last quote, the price is descending
0	Two consecutive quotes without change
1	Compared with last quote, the price is ascending
2	Compared with next- to- last quote, the price is ascending; but compared with last quote, the price has not changed.

Parameters:

iTicktype: Required. A String value. The following constants are available:

Constant	Comment
0	get quote tick

Example

C#:

```
using System;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   public  class  Sample: UserScript
   {
      public override void OnInit()
      {

      }
      public override void OnTick(int index)
      {
         var tick=API.GetTick(0);
          Print("tick:"+tick);
      }
   }
}
```

1.18 GetBarState

Description:

- **GetBarState ()**

Returns a status flag from the EFS engine indicating what bar processing is currently taking place.
The return value is an integer value, as follows:

Return value	Constant	Comment
BARSTATE_NEWBAR	0	the first tick of a new bar has arrived
BARSTATE_CURRENTBAR	1	a new tick has arrived in the current bar
BARSTATE_ALLBARS	2	script is initializing

Parameters:

None, function takes no parameters.

Example

C#:

```
using System;
using  Algo.API;
using  Algo.API.Order;
```

```
namespace Test
{
   public  class  Sample: UserScript
   {
      public override void OnTick(int index)
      {
          int barstate= API.GetBarState();
          Print("the bar state："+ barstate);
      }
   }
}
```

1.19 GetSymbol

Description:

- **GetSymbol ()**

Returns the name of the symbol currently being charted.

Parameters:

None, function takes no parameters.

Example

C#:

```
using System;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   public  class  Sample: UserScript
   {
      public override void OnTick(int index)
      {
          var symbol=API.GetSymbol();
          Print("the sym:"+symbol);
      }
   }
}
```

1.20 BarAsk

Description:

BarAsk ()

Enable the given symbol to receive the quote update event from current bar.

Parameters:

In C#, 'index' equal to OnTick parameter is current bar.

Example

C#

```
using System;
using  Algo.API;
using  Algo.API.Order;

namespace Test
{
   public  class  Sample: UserScript
   {
      public override void OnTick(int index)
      {
      var CurBarAsk=API.BarAsk(index);
      Print("The current bar ASK is :"+ CurBarAsk);
      }
   }
}
```

1.21 BarBid

Description:

BarBid ()

Enable the given symbol to receive the quote update event from current bar.

Parameters:

In C#, 'index' equal to OnTick parameter is current bar.

Example

C#

```
using System;
using  Algo.API;
using  Algo.API.Order;

namespace Test
{
   public  class  Sample: UserScript
   {
      public override void OnTick(int index)
      {
      var CurBarBid=API.BarBid(index);
      Print("The current bar BID is :"+ CurBarBid);
      }
   }
}
```

1.22 GetBarCount

Description:

- **getBarCount()**

Enable the funtion to calculate bars' number by the current time interval of chart on Fortex6UI.

Parameters:

None, function takes no parameters.

C#

```
using System;
using  Algo.API;
using  Algo.API.Order;

namespace Test
{
   public  class  Sample: UserScript
   {
      public override void OnTick(int index)
      {
```

```
    var CurBarBid=API.BarBid(index);
    Print("The current bar BID is :"+ CurBarBid);
     }
  }
}
```

1.23 Getpoint

Description:

- **getPoint(*[,sSymbol]*)**

Returns the value of current symbol 1pip.

Parameters:

sSymbol **:**Optional. The symbol for which to initiate the trade

Example

C#:

```
using System;
using  Algo.API;
using  Algo.API.Order;

namespace Test
{
   public  class  Sample: UserScript
   {
      bool flag = false;
      public override void OnTick(int index)
      {
      if(!false)
      {
      var pip=API.GetPoint();
      Print("The cur symbol 1 pip is :"+ pip);
      flag=true;
      }
      }
   }
}
```

2. Chart data processing

2.1 Price Functions

2.1.1 High

Description:

- **High (***iBarIndex* [,*iNumBars*]**)**

Returns the high price at the specified bar index.

Parameters:

- *iBarIndex:* Required. The Bar index of series to retrieve.
- *iNumBars*: Optional. Number of bars of data to return.

Note：

high (0, 0) not done any processing, returns null.

Example

C#:

```
using System;
using System.Collections.Generic;
using   Algo.API;
using   Algo.API.Order;
using Algo.API.Indicators;
namespace Test
{
    public   class   Sample: UserScript
    {
        public override void OnTick(int index)
        {
            var Highvalue0= MarketSeries.High.GetValue(0);
            Print("The high price of the bar is: " + Highvalue0 );
            List<double> Highvalue= MarketSeries.High.GetValue(0,-4);
            Print("The following datum are 4 bar Highvalue recently: ");
            if(Highvalue.Count>0)
            {
                for(int i=0;i<4;i++)
```

```
                {
                    Print("Highvalue "+ i+":"+ Highvalue[i].ToString());
                }
            }
        }
    }
}
```

2.1.2 Low

Description:

- **Low (***iBarIndex* [,*iNumBars*]**)**

Returns the low price at the specified bar index.

Parameters:

- *iBarIndex:* Required. The Bar index of series to retrieve.
- *iNumBars*: Optional. Number of bars of data to return.

Note：

low (0, 0) not done any processing, returns null.

Example

C#:

```
using System;
using System.Collections.Generic;
using   Algo.API;
using   Algo.API.Order;
using Algo.API.Indicators;
namespace Test
{
    public   class   Sample: UserScript
    {
        public override void OnTick(int index)
        {
            var Lowvalue0= MarketSeries.Low.GetValue(0);
            Print("The Low price of the bar is: " + Lowvalue0 );
            List<double> Lowvalue= MarketSeries.Low.GetValue(0,-4);
            Print("The following datum are 4 bar Lowvalue recently: ");
            if(Lowvalue.Count>0)
```

```
                {
                    for(int i=0;i<4;i++)
                    {
                        Print("Lowvalue "+ i+":"+ Lowvalue[i].ToString());
                    }
                }
            }
        }
    }
```

2.1.3 Close

Description:

- **Close (***iBarIndex* [,*iNumBars*]**)**

Returns the close price at the specified bar index.

Parameters:

- *iBarIndex:* Required. The Bar index of series to retrieve.
- *iNumBars*: Optional. Number of bars of data to return.

Note：

close (0, 0) not done any processing, returns null.

Example

C#:

```
using System;
using System.Collections.Generic;
using   Algo.API;
using   Algo.API.Order;
using Algo.API.Indicators;
namespace Test
{
    public   class   Sample: UserScript
    {
        public override void OnTick(int index)
        {
            var Closevalue0= MarketSeries.Close.GetValue(0);
            Print("The Close price of the bar is: " + Closevalue0 );
            List<double> Closevalue= MarketSeries.Close.GetValue(0,-4);
```

```
                Print("The following datum are 4 bar Closevalue recently: ");
                if(Closevalue.Count>0)
                {
                    for(int i=0;i<4;i++)
                    {
                        Print("Closevalue "+ i+":"+ Closevalue[i].ToString());
                    }
                }
            }
        }
    }
```

2.1.4 Open

Description:

- **Open (***iBarIndex* [,*iNumBars*]**)**

Returns the open price at the specified bar index.

Parameters:

- *iBarIndex:* Required. The Bar index of series to retrieve.
- *iNumBars*: Optional. Number of bars of data to return.

Note：

open (0, 0) not done any processing, returns null.

Example

C#:

```
using System;
using System.Collections.Generic;
using   Algo.API;
using   Algo.API.Order;
using Algo.API.Indicators;
namespace Test
{
    public   class   Sample: UserScript
    {
        public override void OnTick(int index)
        {
            var Openvalue0= MarketSeries.Open.GetValue(0);
```

```
                Print("The Open price of the bar is: " + Openvalue0 );
                List<double> Openvalue= MarketSeries.Open.GetValue(0,-4);
                Print("The following datum are 4 bar Openvalue recently: ");
                if(Openvalue.Count>0)
                {
                    for(int i=0;i<4;i++)
                    {
                        Print("Openvalue "+ i+":"+ Openvalue[i].ToString());
                    }
                }
            }
        }
    }
```

2.1.5 Highest

Description:

- **Highest (*iNumbars, iSeries*)**

This function will return the highest value found in a series within numBars number of bars.

Parameters:

- *iNumbars:* Required. The number of bars back to search.
- *iSeries:* Required. The series in which to search for the value.

Example

C#:

```
using System;
using System.Collections.Generic;
using   Algo.API;
using   Algo.API.Order;
using Algo.API.Indicators;

    namespace Test
    {

   public  class  Sample: UserScript
   {
```

```
    public override void OnTick(int index)
    {
    List<double> Highvalue= MarketSeries.High.GetValue(0,-5);
                    Print("The following datum are 5 bar Highvalue recently: ");
                    if(Highvalue.Count>0)
                    {
                        for(int i=0;i<5;i++)
                        {
                            Print("Highvalue "+ i+":"+ Highvalue[i].ToString());
                        }
                    }
        Print("find the highest-high over the last 4 bars ");
        double val= IndicatorHelper.Highest(5,MarketSeries.High);
        Print(val.ToString());
    }
  }
}
```

2.1.6 hhv

Description:

- **hhv (*iNumbars, iSeries*)**

This function will return the highest value found in a series within numBars number of bars.

Parameters:

- *iNumbars:* Required. The number of bars back to search.
- *iSeries:* Required. The series in which to search for the value.

Example

C#:

```
No 'hhv' to get highest value in C#.
```

2.1.7 llv

Description:

- **llv (*iNumbars, iSeries*)**

This function will return the lowest value found in a series within numBars number of bars.

Parameters:

iNumbars: Required. The number of bars back to search.
iSeries: Required. The series in which to search for the value.

Example

C#:

No 'llv' to get Lowest value in C#.

2.1.8 Lowest

Description:

- **Lowest (***iNumbars, iSeries***)**

This function will return the lowest value found in a series within numBars number of bars.

Parameters:

- *iNumbars:* Required. The number of bars back to search.
- *iSeries:* Required. The series in which to search for the value.

Example

C#:

```
using System;
using System.Collections.Generic;
using    Algo.API;
using    Algo.API.Order;
using Algo.API.Indicators;

    namespace Test
    {

   public  class  Sample: UserScript
   {



      public override void OnTick(int index)
```

```
    {
    List<double> Lowvalue= MarketSeries.Low.GetValue(0,-5);
                    Print("The following datum are 5 bar Lowvalue recently: ");
                    if(Lowvalue.Count>0)
                   {
                        for(int i=0;i<5;i++)
                        {
                            Print("Lowvalue "+ i+":"+ Lowvalue[i].ToString());
                        }
                    }
        Print("find the Lowest-Low over the last 4 bars ");
        double val= IndicatorHelper.Lowest(5,MarketSeries.Low);
        Print(val.ToString());
    }
  }
}
```

2.2 Time Functions

2.2.1 Year

Description:

- **Year (***iBarIndex* [,*iNumBars*]**)**

Returns the year at the specified bar index.

Parameters:

- *iBarIndex:* Required. The Bar index of series to retrieve.
- *iNumBars*: Optional. Number of bars of data to return.

Note：

Year (0, 0) not done any processing, returns null.

Example

C#:

```
using System;
using System.Collections.Generic;
using  Algo.API;
using Algo.API.Indicators;
using  Algo.API.Date;
```

```
namespace Test
{
   public  class  Sample: UserScript
   {
      public override void OnTick(int index)
      {
           if (! API.IsHistoryBar ())
           {
                //retrieve the value for the current price bar
                var strYear1 =BarDate.Year.GetValue(0);
                Print ("current year: " + strYear1);
           }
           List<int> year= BarDate.Year.GetValue(0,-4);
         Print("retrieve the last 4 values into an array");
         if(year.Count>0)
         {
           for(int i=0;i<4;i++)
           {
             Print("Last Year_Bar"+i+":"+year[i].ToString());
           }
         }
      }
   }
}
```

2.2.2 Month

Description:

- **Month (***iBarIndex* [,*iNumBars*]**)**

Returns the month at the specified bar index.

Parameters:

- *iBarIndex:* Required. The Bar index of series to retrieve.
- *iNumBars*: Optional. Number of bars of data to return.

Note：

month (0, 0) not done any processing, returns null.

Example

C#:

```
using System;
using System.Collections.Generic;
using  Algo.API;
using Algo.API.Indicators;
using  Algo.API.Date;

namespace Test
{
   public  class  Sample: UserScript
   {
      public override void OnTick(int index)
      {
           if (! API.IsHistoryBar ())
           {
                //retrieve the value for the current price bar
                var strMonth1 =BarDate.Month.GetValue(0);
                Print ("current Month: " + strMonth1);
           }
           List<int> Month= BarDate.Month.GetValue(0,-4);
        Print("retrieve the last 4 values into an array");
        if(Month.Count>0)
        {
          for(int i=0;i<4;i++)
          {
            Print("Last Month_Bar"+i+":"+Month[i].ToString());
          }
        }
      }
   }
}
```

2.2.3 Day

Description:

- **Day (***iBarIndex* [,*iNumBars*]**)**

Returns the day at the specified bar index.

Parameters:

- *iBarIndex:* Required. The Bar index of series to retrieve.
- *iNumBars*: Optional. Number of bars of data to return.

Note：

Day (0, 0) not done any processing, returns null.

Example

C#:

```
using System;
using System.Collections.Generic;
using  Algo.API;
using Algo.API.Indicators;
using  Algo.API.Date;

namespace Test
{
   public  class  Sample: UserScript
   {
      public override void OnTick(int index)
      {
           if (! API.IsHistoryBar ())
           {
                //retrieve the value for the current price bar
                var strDay1 =BarDate.Day.GetValue(0);
                Print ("current day: " + strDay1);
           }
           List<int> day= BarDate.Day.GetValue(0,-4);
        Print("retrieve the last 4 values into an array");
        if(day.Count>0)
        {
          for(int i=0;i<4;i++)
          {
            Print("Last Day_Bar"+i+":"+day[i].ToString());
          }
        }
      }
   }
}
```

2.2.4 Hour

Description:

- **Hour (***iBarIndex* [,*iNumBars*])

Returns the hour at the specified bar index.

Parameters:

- *iBarIndex:* Required. The Bar index of series to retrieve.
- *iNumBars*: Optional. Number of bars of data to return.

Note：

Hour (0, 0) not done any processing, returns null.

Example

C#:

```
using System;
using System.Collections.Generic;
using  Algo.API;
using Algo.API.Indicators;
using  Algo.API.Date;

namespace Test
{
   public  class  Sample: UserScript
   {
      public override void OnTick(int index)
      {
           if (! API.IsHistoryBar ())
           {
                //retrieve the value for the current price bar
                var strHour1 =BarDate.Hour.GetValue(0);
                Print ("current Hour: " + strHour1);
           }
           List<int> Hour= BarDate.Hour.GetValue(0,-4);
        Print("retrieve the last 4 values into an array");
        if(Hour.Count>0)
        {
          for(int i=0;i<4;i++)
          {
            Print("Last Hour_Bar"+i+":"+Hour[i].ToString());
          }
        }
      }
   }
}
```

2.2.5 Minute

Description:

- **Minute (***iBarIndex* [,*iNumBars*]**)**

Returns the minute at the specified bar index.

Parameters:

- *iBarIndex:* Required. The Bar index of series to retrieve.
- *iNumBars*: Optional. Number of bars of data to return.

Note：

minute (0, 0) not done any processing, returns null.

Example

C#:

```
using System;
using System.Collections.Generic;
using  Algo.API;
using Algo.API.Indicators;
using  Algo.API.Date;

namespace Test
{
   public  class  Sample: UserScript
   {
      public override void OnTick(int index)
      {
           if (! API.IsHistoryBar ())
           {
                //retrieve the value for the current price bar
                var strMinute1 =BarDate.Minute.GetValue(0);
                Print ("current Minute: " + strMinute1);
           }
           List<int> Minute= BarDate.Minute.GetValue(0,-4);
        Print("retrieve the last 4 values into an array");
        if(Minute.Count>0)
        {
          for(int i=0;i<4;i++)
          {
```

```
            Print("Last Minute_Bar"+i+":"+Minute[i].ToString());
          }
        }
      }
    }
}
```

2.2.6 Second

Description:

- **Second (***iBarIndex* [,*iNumBars*]**)**

Returns the second at the specified bar index.

Parameters:

- *iBarIndex:* Required. The Bar index of series to retrieve.
- *iNumBars*: Optional. Number of bars of data to return.

Note：

Second (0, 0) not done any processing, returns null.

Example

C#:

```
using System;
using System.Collections.Generic;
using  Algo.API;
using Algo.API.Indicators;
using  Algo.API.Date;

namespace Test
{
   public  class  Sample: UserScript
   {
      public override void OnTick(int index)
      {
          if (! API.IsHistoryBar ())
          {
                //retrieve the value for the current price bar
                var strSecond1 =BarDate.Second.GetValue(0);
                Print ("current Second: " + strSecond1);
```

```
            }
            List<int> Second= BarDate.Second.GetValue(0,-4);
          Print("retrieve the last 4 values into an array");
          if(Second.Count>0)
          {
            for(int i=0;i<4;i++)
            {
              Print("Last Second_Bar"+i+":"+Second[i].ToString());
            }
          }
        }
    }
}
```

2.2.7 GetYear

Description:

- **GetYear (***iBarIndex* [,*iNumBars*]**)**

Returns the year at the specified bar index.

Parameters:

- *iBarIndex:* Required. The Bar index of series to retrieve.
- *iNumBars*: Optional. Number of bars of data to return.

Note：

getYear (0, 0) not done any processing, returns null.

Example

C#:

```
No 'getYear' to get Year value in C#.
```

2.2.8 GetMonth

Description:

- **GetMonth (***iBarIndex* [,*iNumBars*]**)**

Returns the month at the specified bar index.

Parameters:

- *iBarIndex:* Required. The Bar index of series to retrieve.
- *iNumBars*: Optional. Number of bars of data to return.

Note：

GetMonth (0, 0) not done any processing, returns null.

Example

C#:

```
No 'getMonth' to get month value in C#.
```

2.2.9 GetDay

Description:

- **GetDay (***iBarIndex* [,*iNumBars*]**)**

Returns the day at the specified bar index.

Parameters:

- *iBarIndex:* Required. The Bar index of series to retrieve.
- *iNumBars*: Optional. Number of bars of data to return.

Note：

getDay (0, 0) not done any processing, returns null.

Example

C#:

```
No 'getDay' to get Day value in C#.
```

2.2.10 GetMinute

Description:

- **GetMinute (***iBarIndex* [,*iNumBars*])

Returns the minute at the specified bar index.

Parameters:

- *iBarIndex:* Required. The Bar index of series to retrieve.
- *iNumBars*: Optional. Number of bars of data to return.

Note：

GetMinute (0, 0) not done any processing, returns null.

Example

C#：

No 'getMinute' to get Minute value in C#.

2.2.11 GetSecond

Description:

- **GetSecond (***iBarIndex* [,*iNumBars*])

Returns the second at the specified bar index.

Parameters:

- *iBarIndex:* Required. The Bar index of series to retrieve.
- *iNumBars*: Optional. Number of bars of data to return.

Note：

GetSecond (0, 0) not done any processing, returns null.

Example

C#:

No 'getSecond' to get second value in C#.

3. Built-In Study Functions

3.1 Average True Range

Description:

- **atr (***length***)**

The **Average True Range** (ATR) is a measure of volatility. It was introduced by Welles Wilder in his book New Concepts in Technical Trading Systems and has since been used as a component of many indicators and trading systems.
Wilder has found that the high ATR values often occur at market bottoms following a "panic" sell-off. Low ATR values are often found during extended sideways periods, such as those found at tops and after consolidation periods.

Parameters:

length: Required, period to use for the calculation.

Example

C#:

```
using System;
using Algo.API;
using Algo.API.Drawing;
using Algo.API.Indicators;
 using Algo.API.Interface;

namespace Test
{

    public class Sample : UserScript
    {

        [Output("ATR", Color = Colors.Yellow)]
        private IDataSeries ds = new DataSeries();

        [Parameter("Length",DefaultValue=14)]
        private int Length{get;set;}

        private IATRStudy atr ;

        public override void OnInit()
        {
            atr = IndicatorAccessor.ATR(Length, MAStudyType.Simple);
            this.SetTitle(string.Format("ATR({0})",Length.ToString()));
            API.SetPriceStudy(false);

        }

        public override void OnCalculate(int index)
        {
```

```
                ds[index] = atr.Result[index];
            }

        }
    }
```

Description:

- **upperBB (***length*, *stdDev* [, *source*] [, *InitIndex*]**)**
- **middleBB (***length*, *stdDev* [, *source*] [, *InitIndex*]**)**
- **lowerBB (***length*, *stdDev* [, *source*] [, *InitIndex*]**)**

3.2 Bollinger Bands

Bollinger Bands (created by John Bollinger) are similar to moving average envelopes. The difference between Bollinger Bands and envelopes is that envelopes are plotted at a fixed percentage above and below a moving average, whereas Bollinger Bands are plotted at standard deviation levels above and below a moving average. Since standard deviation is a measure of volatility, the bands are self-adjusting, widening during volatile markets and contracting during calmer periods.

Parameters:

- *length:* Required. The period to use for the calculation.
- *stdDev:* Required. The number of standard deviations.
- *source*: Optional. Input series for the study.
  Default: close
- *InitIndex:* Optional. The bar index of value to retrieve.
  Default: 0, current value

Example

C#:

```
using System;
using Algo.API;
using Algo.API.Drawing;
using Algo.API.Indicators;
 using Algo.API.Interface;

namespace Test
{
```

```
        public class Sample : UserScript
        {

            [Output("Upper", Color = Colors.Blue)]
            private IDataSeries dsUpper = new DataSeries();

            [Output("Middle", Color = Colors.Red)]
            private IDataSeries dsMiddle = new DataSeries();

            [Output("Lower", Color = Colors.Yellow)]
            private IDataSeries dsLower = new DataSeries();

            [Parameter("Length",DefaultValue=20)]
            private int Length{     get;set;}

            [Parameter("StdDev",DefaultValue=2)]
            private int StdDev{     get;set;}

           [Parameter("Price Source")]
            private IDataSeries Source { get; set; }


            private IBollingerStudy band;


            public override void OnInit()
            {
                band                                                                    =
IndicatorAccessor.Bollinger(Length,StdDev,Source ,MAStudyType.Simple);
                this.SetTitle(string.Format("Bollinger({0},{1})",Length,StdDev));
            }

            public override void OnCalculate(int index)
            {
                dsUpper[index] = band.UpperBB[index];
                dsMiddle[index] = band.MiddleBB[index];
                dsLower[index] = band.LowerBB[index];
            }

        }
    }
```

3.3 Commodity Channel Index

Description:

- **cci (***length* [, *source*] [, *InitIndex*]**)**

The **Commodity Channel Index** (CCI) is a price momentum indicator that measures the price excursions from the mean price as a statistical variation. It is used to detect the beginnings and endings of trends.

Parameters:

- *length:* Required. The period to use for the calculation.
- *source* Optional. Input series for the study.
  Default: close
- *InitIndex:* Optional. The bar index of value to retrieve.
  Default: 0, current value。

Example

C#:

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Text;
using Algo.API;
using Algo.API.Drawing;
using Algo.API.Indicators;
using Algo.API.Interface;

namespace Algo.API
{
    public class Sample : UserScript
    {

        [Output("CCI", Color = Colors.Blue)]
        private IDataSeries ds = new DataSeries();

        [Parameter("Length",DefaultValue=20)]
        private int Length{get;set;}

        [Parameter("Price Source")]
        private IDataSeries Source { get; set; }
```

```
        private ICCIStudy cci;


        public override void OnInit()
        {

            cci = IndicatorAccessor.Cci(Length, Source);
            API.SetPriceStudy(false);
            this.SetTitle(string.Format("CCI({0})",Length.ToString()));
        }

        public override void OnCalculate(int index)
        {
            ds[index] = cci.Result[index];

        }

    }

}
```

3.4 Moving Average

Description:

- **ema (***length* [, *source*] [, *InitIndex*]**)**
- **sma (***length* [, *source*] [, *InitIndex*]**)**
- **wma (***length* [, *source*] [, *InitIndex*]**)**
- **vwma (***length* [, *source*] [, *InitIndex*]**)**

A **Moving Average** is an indicator that shows the average value of a security's price over a period of time. When calculating a moving average, a mathematical analysis of the security's average value over a predetermined time period is made. As the security's price changes, its average price moves up or down.

Parameters:

- *length:* Required. The period to use for the calculation.
- *source*: Optional. Input series for the study.
  Default: close
- *InitIndex:* Optional. The bar index of value to retrieve.
  Default: 0, current value.

Example

C#:

```
using System;
using Algo.API;
using Algo.API.Drawing;
using Algo.API.Indicators;
 using Algo.API.Interface;

namespace Test
{

    public class Sample : UserScript
    {

        [Output("MA", Color = Colors.Yellow)]
        private IDataSeries ds = new DataSeries();

        [Parameter("Length",DefaultValue=10)]
        private int Length{get;set;}

        [Parameter("Price Source")]
        private IDataSeries Source { get; set; }

        private IMAStudy ma ;

        public override void OnInit()
        {
ma=IndicatorAccessor.Ma(Length, Source,MAStudyType.Exponential);
            this.SetTitle(string.Format("Ma({0})",Length.ToString()));

        }

        public override void OnCalculate(int index)
        {
            ds[index] = ma.Result[index];
        }
```

```
        }
    }
```

3.5 MACD

Description:

- **macd (***fastLength*, *slowLength*, *smoothing* [, *source*] [, *barIndex*]**)**
- **macdSignal (***fastLength*, *slowLength*, *smoothing* [, *source*] [, *barIndex*]**)**
- **macdHist (***fastLength*, *slowLength*, *smoothing* [, *source*] [, *barIndex*]**)**

MACD is short for Moving Average Convergence Divergence. The MACD looks at the difference between a short-term moving average and a long-term moving average. A third moving average is taken of the difference and used as a signal line.

Parameters:

- *fastLength: Required. the fast MACD period.*
- *slowLength: Required. the slow MACD period.*
- *smoothing: the MACD smoothing period.*
- *source: Optional. Input series for the study.*
  Default: close
- *InitIndex: Optional. The bar index of value to retrieve.*
  Default: 0, current value

Example

C#:

```
using System;
using Algo.API;
using Algo.API.Indicators;
using Algo.API.Drawing;
using Algo.API.Interface;
 using Algo.API.Interface;

namespace Test
{
    //the base class of trade operation
    public class Sample : UserScript
    {
```

```
                [Output("Macd", Color = Colors.Blue)]
                IDataSeries dsMacd = new DataSeries();

                [Output("Single", Color = Colors.Red)]
                IDataSeries dsSingal = new DataSeries();

                [Output("Hist", Color = Colors.Yellow, LineType = LineType.Bar)]
                IDataSeries dsHist = new DataSeries();

                [Output("Zero",Color = Colors.White)]
                IDataSeries dsZero = new DataSeries();


                [Parameter("Fast",DefaultValue=12)]
                private int Fast{     get;set;}

                [Parameter("Slow",DefaultValue=26)]
                private int Slow{     get;set;}

                [Parameter("Smoothing",DefaultValue=9)]
                private int Smoothing{     get;set;}

                 [Parameter("Price Source")]
                private IDataSeries Source { get; set; }

                private IMACDStudy macd ;


                public override void OnInit()
                {
                      macd   =   IndicatorAccessor.Macd(Fast,   Slow,   Smoothing,
Source);

this.SetTitle(string.Format("MACD({0},{1},{2})",Fast,Slow,Smoothing));
                      AlgoAPI.GetAPIObject().SetPriceStudy(false);

                }

                public override void OnCalculate(int index)
                {
                      dsMacd[index] = macd.Macd[index];
                      dsSingal[index] = macd.Singal[index];
                      dsHist[index] = macd.Hist[index];
```

```
                    dsZero[index] = 0;


                }

            }
        }
```

3.6 Rate of Change

Description:

- **roc (***length* [, *source*] [, *InitIndex*]**)**

The **Rate-of-Change** (ROC) indicator displays the difference between the current price and the price x time periods ago. The difference can be displayed either in points or as a percentage. The momentum indicator displays the same information but expresses it as a ratio.

Parameters:

- *length: Required. The period to use for the calculation.*
- *source: Optional. Input series for the study.*
  Default: close
- *InitIndex: Optional. The bar index of value to retrieve.*
  Default: 0, current value.

Example

C#:

```
using System;
using System;
using Algo.API;
using Algo.API.Drawing;
using Algo.API.Indicators;
 using Algo.API.Interface;

namespace Test
{

    public class Sample : UserScript
    {
```

```
        [Output("ROC", Color = Colors.Red)]
        private IDataSeries ds = new DataSeries();

        [Parameter("Length",DefaultValue=9)]
        private int Length{     get;set;}

         [Parameter("Price Source")]
        private IDataSeries Source { get; set; }

        private IROCStudy roc;

        public override void OnInit()
        {
              roc = IndicatorAccessor.Roc(Length,Source);
            this.SetTitle(string.Format("ROC({0})",Length));
            API.SetPriceStudy(false);
        }

        public override void OnCalculate(int index)
        {
            ds[index] = roc.Result[index];
        }


    }
}
```

4. Trading Functions

4.1 BuyMarket

Description:

- **BuyMarket(***iQuantity [,sSymbol] [,dSL][,dTP]*[,*sAccountName*] [,*sExpiry*]**)**

Submit a buy market order to broker.

Parameters:

- *iQuantity*: Required, a positive integer value. The number of shares/contracts.

- *dSL: Optional,StopLoss price.*
- *dTP: Optional,TakeProfit price.*
- *sAccountName:* Optional. The account name for which to initiate the trade.
- *sSymbol:* Optional. The symbol for which to initiate the trade
- *sExpiry:* Optional. Expiration for the order. The following constants are available:

Constant	Comment
FOK	TIF Setting is “FOK”
GTC	TIF Setting is “GTC”
IOC	TIF Setting is “IOC”

Example ：

C#:

```
using System;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   //the base class of trade operation
   public  class  Sample:UserScript
   {
      bool flag = false;
      public override void OnInit()
      {
        API.EnableTrading();
      }
       public override void OnTick(int index)
      {
          bool isHistory = API.IsHistoryBar();
          if(!isHistory && !flag)
          {
            OrderRequest order = new OrderRequest();
            order.BuyMarket(10000);
            flag= true;
           }
      }
   }
}
```

4.2 BuyLimit

Description:

BuyLimit (*iQuantity*, *dLimitPrice [,sSymbol] [,dSL][,dTP][,sAccountName] [,sExpiry]***)**

Submit a buy limit order to broker.

Parameters:

- *iQuantity*: Required, a positive integer value. The number of shares/contracts.
- *dSL: Optional,StopLoss price.*
- *dTP: Optional,TakeProfit price.*
- *dLimitPrice*: Required. The limit price.
- *sAccountName:* Optional. The account name for which to initiate the UserScript.
- *sSymbol:* Optional. The symbol for which to initiate the trade
- *sExpiry:* Optional. Expiration for the order. The following constants are available:

Constant	Comment
FOK	TIF Setting is “FOK”
GTC	TIF Setting is “GTC”
IOC	TIF Setting is “IOC”

Example ：

C#:

```
using System;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   //the base class of trade operation
   public  class  Sample:UserScript
   {
      bool flag = false;
      public override void OnInit()
      {
        API.EnableTrading();
      }
      public override void OnTick(int index)
      {
          bool isHistory = API.IsHistoryBar();
                  if(!isHistory && !flag)
          {
            var limit_price = API.GetQuote ("ASK") - 0.02;
            Print("limit_price:"+limit_price);
            OrderRequest order = new OrderRequest();
            order.BuyLimit(10000,limit_price);
```

```
            flag= true;
          }
      }
   }
}
```

4.3 BuyStop

Description:

- **BuyStop (***iQuantity*, *dStopPrice* [,*sSymbol*] *[,dSL][,dTP]*[,*sAccountName*] [,*sExpiry*]**)**

Submit a buy stop order to broker.

Parameters:

- *iQuantity*: Required, a positive integer value. The number of shares/contracts.
- *dStopPrice:* Required. The stop price.
- *dSL: Optional,StopLoss price.*
- *dTP: Optional,TakeProfit price.*
- *sAccountName:* Optional. The account name for which to initiate the trade.
- *sSymbol:* Optional. The symbol for which to initiate the trade
- *sExpiry:* Optional. Expiration for the order. The following constants are available:

Constant	Comment
FOK	TIF Setting is “FOK”
GTC	TIF Setting is “GTC”
IOC	TIF Setting is “IOC”

Example：

C#:

```
using System;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   //the base class of trade operation
   public  class  Sample:UserScript
   {
      bool flag = false;
      public override void OnInit()
      {
        API.EnableTrading();
```

```
    }
    public override void OnTick(int index)
    {
        bool isHistory = API.IsHistoryBar();
        var Stop_price = API.GetQuote ("ASK") + 0.02;
        Print("Stop_price:"+Stop_price);
        if(!isHistory && !flag)
        {
          var Stop_price = API.GetQuote ("ASK") + 0.02;
          Print("Stop_price:"+Stop_price);
          OrderRequest order = new OrderRequest();
          order.BuyStop(10000,Stop_price);
          flag= true;
        }
    }
  }
}
```

4.4 BuyThreshold

Description:

- **BuyThreshold (***iQuantity*, *dLowerlimit*, *dUpperLimit [,sSymbol]* [,*sAccountName*] [,*sExpiry*]**)**

Submit a buy threshold order to broker.

Parameters:

- *iQuantity*: Required, a positive integer value. The number of shares/contracts.
- *dLowerlimit:* Required. The lower price.
- *dUpperLimit:* Required. The upper price.
- *sAccountName:* Optional. The account name for which to initiate the trade.
- *sSymbol:* Optional. The symbol for which to initiate the trade
- *sExpiry:* Optional. Expiration for the order. The following constants are available:
-

Constant	Comment
FOK	TIF Setting is “FOK”
GTC	TIF Setting is “GTC”
IOC	TIF Setting is “IOC”

Example

C#:

```
using System;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   //the base class of trade operation
   public  class  Sample:UserScript
   {
      bool flag = false;
      public override void OnInit()
      {
        API.EnableTrading();
      }
      public override void OnTick(int index)
      {
          bool isHistory = API.IsHistoryBar();

          if(!isHistory && !flag)
          {
            var UpperBound = API.GetQuote ("ASK") +0.02;
            var LowerBound= API.GetQuote ("ASK") -0.02;
            Print("UpperBound:"+ UpperBound+" LowerBound:"+LowerBound);
            OrderRequest order = new OrderRequest();
            order.BuyThreshold(10000,LowerBound,UpperBound);
            flag= true;
           }
      }
   }
}
```

4.5 Buy

Description:

- **Buy (*[sAccountName] , [sSymbol]*)**

Submit a market order to broker using the pre-set default number of shares/contracts.

Parameters:

- *sAccountName:* Optional. The account name for which to initiate the trade.
- *sSymbol:* Optional. The symbol for which to initiate the trade

Example ：

C#：

```
using System;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   //the base class of trade operation
   public  class  Sample:UserScript
   {
      bool flag = false;
      public override void OnInit()
      {
        API.EnableTrading();
        var intQuan = API.GetDefaultQuantity();
        if(intQuan<1000){
        API.SetDefaultQuantity(10000);
      }
      }
      public override void OnTick(int index)
      {
          bool isHistory = API.IsHistoryBar();
          if(!isHistory && !flag)
          {
            OrderRequest order = new OrderRequest();
            order.Buy();
            flag= true;
          }
      }
   }
}
```

4.6 sellMarket

Description:

- **sellMarket (***iQuantity, [,sSymbol] [,dSL][,dTP][,sAccountName] [,sExpiry]***)**

Submit a sell market order to broker.

Parameters:

- *iQuantity*: Required, a positive integer value. The number of shares/contracts.
- *dSL: Optional,StopLoss price.*

- *dTP: Optional,TakeProfit price.*
- *sAccountName:* Optional. The account name for which to initiate the trade.
- *sSymbol:* Optional. The symbol for which to initiate the trade.
- *sExpiry:* Optional. Expiration for the order. The following constants are available:

Constant	Comment
FOK	TIF Setting is “FOK”
GTC	TIF Setting is “GTC”
IOC	TIF Setting is “IOC”

Example

C#:

```
using System;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   //the base class of trade operation
   public  class  Sample:UserScript
   {
      bool flag = false;
      public override void OnInit()
      {
        API.EnableTrading();
      }
       public override void OnTick(int index)
      {
          bool isHistory = API.IsHistoryBar();
          if(!isHistory && !flag)
          {
            OrderRequest order = new OrderRequest();
            order.SellMarket(10000);
            flag= true;
           }
      }
   }
}
```

4.7 SellLimit

Description:

- **SellLimit(***iQuantity,dLimitPrice[,sSymbol] [,dSL][,dTP][,sAccountName]*

[,sExpiry])

Submit a sell limit order to broker.

Parameters:

- *iQuantity*: Required, a positive integer value. The number of shares/contracts.
- *dLimitPrice*: Required. The limit price.
- *dSL: Optional,StopLoss price.*
- *dTP: Optional,TakeProfit price.*
- *sAccountName:* Optional. The account name for which to initiate the trade.
- *sSymbol:* Optional. The symbol for which to initiate the trade
- *sExpiry:* Optional. Expiration for the order. The following constants are available:

Constant	Comment
FOK	TIF Setting is “FOK”
GTC	TIF Setting is “GTC”
IOC	TIF Setting is “IOC”

Example

C#:

```
using System;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   //the base class of trade operation
   public  class  Sample:UserScript
   {
      bool flag = false;
      public override void OnInit()
      {
        API.EnableTrading();
      }
      public override void OnTick(int index)
      {
          bool isHistory = API.IsHistoryBar();
          if(!isHistory && !flag)
          {
              var limit_price = API.GetQuote ("BID") + 0.02;
               Print("limit_price:"+limit_price);
              OrderRequest order = new OrderRequest();
             order.SellLimit(10000,limit_price);
              flag= true;
```

```
        }
    }
  }
}
```

4.8 SellStop

Description:

- **SellStop(***iQuantity,dStopPrice[,sSymbol][,dSL][,dTP][,sAccountName][,sExpiry]***)**

Submit a sell stop order to broker.

Parameters:

- *iQuantity*: Required, a positive integer value. The number of shares/contracts.
- *dStopPrice:* Required. The stop price.
- *dSL: Optional,StopLoss price.*
- *dTP: Optional,TakeProfit price.*
- *sAccountName:* Optional. The account name for which to initiate the trade.
- *sSymbol:* Optional. The symbol for which to initiate the trade
- *sExpiry:* Optional. Expiration for the order. The following constants are available:

Constant	Comment
FOK	TIF Setting is “FOK”
GTC	TIF Setting is “GTC”
IOC	TIF Setting is “IOC”

Example

C#:

```
using System;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   //the base class of trade operation
   public  class  Sample:UserScript
   {
      bool flag = false;
      public override void OnInit()
      {
        API.EnableTrading();
      }
```

```
        public override void OnTick(int index)
        {
            bool isHistory = API.IsHistoryBar();
            if(!isHistory && !flag)
            {
                var Stop_price = API.GetQuote ("BID") - 0.02;
                Print("Stop_price:"+Stop_price);
                OrderRequest order = new OrderRequest();
                order.SellStop(10000,Stop_price);
                flag= true;
            }
        }
    }
}
```

4.9 SellThreshold

Description:

- **SellThreshold** *(iQuantity, dLowerlimit, dUpperLimit [,sAccountName] [,sSymbol] [,sExpiry])*

Submit a sell threshold order to broker.

Parameters:

- *iQuantity*: Required, a positive integer value. The number of shares/contracts.
- *dLowerlimit:* Required. The lower price.
- *dUpperLimit:* Required. The upper price.
- *sAccountName:* Optional. The account name for which to initiate the trade.
- *sSymbol:* Optional. The symbol for which to initiate the trade
- *sExpiry:* Optional. Expiration for the order. The following constants are available:

Constant	Comment
FOK	TIF Setting is “FOK”
GTC	TIF Setting is “GTC”
IOC	TIF Setting is “IOC”

Example

C#:

```
using System;
using  Algo.API;
using  Algo.API.Order;
namespace Test
```

```
{
   //the base class of trade operation
   public  class  Sample:UserScript
   {
      bool flag = false;
      public override void OnInit()
      {
        API.EnableTrading();
      }
      public override void OnTick(int index)
      {
          bool isHistory = API.IsHistoryBar();
          if(!isHistory && !flag)
          {
              var UpperBound = API.GetQuote ("bid") +0.02;
              var LowerBound= API.GetQuote ("bid") -0.02;
              Print("UpperBound:"+ UpperBound +" LowerBound:"+LowerBound);
              OrderRequest order = new OrderRequest();
              order.SellThreshold(10000,LowerBound,UpperBound);
                  flag= true;
                      }
                  }
              }
          }
```

4.10 Sell

Description:

- **Sell (**[*sAccountName*] [,*sSymbol*]**)**

Submit a market order to broker using the pre-set default number of shares/contracts.

Parameters:

- *sAccountName:* Optional. The account name for which to initiate the trade.
- *sSymbol:* Optional. The symbol for which to initiate the trade.

Example

C#:

```
using System;
using  Algo.API;
using  Algo.API.Order;
```

```
namespace Test
{
   //the base class of trade operation
   public  class  Sample:UserScript
   {
      bool flag = false;
      public override void OnInit()
      {
        API.EnableTrading();
        var intQuan = API.GetDefaultQuantity();
        if(intQuan<1000){
        API.SetDefaultQuantity(10000);
       }
     }
      public override void OnTick(int index)
      {
          bool isHistory = API.IsHistoryBar();
          if(!isHistory && !flag)
          {
            OrderRequest order = new OrderRequest();
            order.Sell();
            flag= true;
           }
      }
   }
}
```

4.11 SendOrder

Description:

- **SendOrder** *(sSide, sOrderType, iQuantity [,sSymbol] [,dStopPrice] [,dLimitPrice] [,dSL] [,dTP] [,sAccountName] [,sExpiry] [,sMMID] [, sFillType])*

Submit an order with all parameter to broker.

Parameters:

- *sSide:* Required. A String value. The following constants are available:

Constant	Comment
buy	Submit a buy order.
sell	Submit a sell order.

- *sOrderType:* Required. A String value. The following constants are available:

Constant	Comment
market	Submit a market order.
limit	Submit a limit order.
stop	Submit a stop order.
threshold	Submit a threshold order.

- *iQuantity*: Required, a positive integer value. The number of shares/contracts.
- *sSymbol:* Optional. The symbol for which to initiate the trade.
- *dStopPrice:* Optional. See below: “stopPrice, limitPrice Explain”.
- *dLimitPrice:* Optional. See below: “stopPrice, limitPrice Explain”.
- *dSL:* Optional. Stop Loss.
- *dTP:* Optional. Take Profit.
- *sAccountName:* Optional. The account name for which to initiate the trade.
- *sExpiry:* Optional. Expiration for the order. The following constants are available:

Constant	Comment
GTC	TIF Setting is “GTC”
IOC	TIF Setting is “IOC”
FOK	TIF Setting is “FOK”

- *sMMID:* Optional.
- *sFillType:* Optional. The following constants are available:

Constant	Comment
PART	default
AON	

dStopPrice, dLimitPrice Explain：

sOrderType	dStopPrice	dLimitPrice
market	Invalid	Invalid
stop	Stop Price	Invalid
limit	Invalid	Limit Price
“Threshold”	Lower limit	Upper limit

Example :

C#：

```
using System;
using   Algo.API;
using   Algo.API.Order;
namespace Test
{
    //the base class of trade operation
    public   class   Sample:UserScript
    {
        bool flag = false;
        public override void OnInit()
```

```
            {
                API.EnableTrading();
              var intQuan = API.GetDefaultQuantity();
            }
            Public override void OnTick(int index)
            {
                    bool isHistory = API.IsHistoryBar();
                    if(!isHistory && !flag)
                    {
                       var upper = API.GetQuote("bid") + 0.02;
                       var lower = API.GetQuote("bid") - 0.05;
                       OrderRequest order = new OrderRequest ();
                                order.SendOrder("sell","threshold",10000,"CAD/JPY",
lower,upper,0,0, "","","","","");
                         flag= true;
                      }
            }
        }
    }
```

5. Account and Portfolio Functions

5.1 Account Object

Description:

Account Object includes some properties and methods about account information. There are two methods for obtaining **Account** object:

1. We can use **getCurrentAccount** function to get the Account object, as follows:

```
var account = getCurrentAccount ();
```

Note: This method only obtains the account object that is currently logged on Fortex.

2. We can use **new** method to create the Account object, as follows:

```
var acc= new Account (AccountID);
```

Note: AccountID is account name. This method can obtain any specified account object.

If the return value is 0 or a negative integer value in the Account Object, it means the function fails to retrieve the quote.

Function List:

Account object have the following functions:

NO,	Functions	Comment

1	**GetAccountName** ()	Retrieve the Account Name
2	**GetPosition** ()	Retrieve the amount of Position
3	**GetOpenPL** ()	Retrieve the value of OpenPL
4	**GetClosePL** ()	Retrieve the value of ClosePL
5	**GetPL** ()	Retrieve the value of PL
6	**GetQty** ()	Retrieve the value of QTY
7	**GetTickets** ()	Retrieve the amount of Tickets
8	**GetAvailableMargin** ()	Retrieve the value of Available_Margin
9	**GetAssetBySymbolName**([*symbolName*])	Retrieve the Asset by symbol name

5.1.1 GetAccountName

Description:

- **GetAccountName ()**

Retrieve the Account Name.

Parameters:

None, function takes no parameters.

Example

C#:

```
using System;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   //the base class of trade operation
   public  class  Sample: UserScript
   {
      public  Sample()
      {
         var account = Account.GetCurrentAccount();
         string name=account.GetAccountName();
         Print(name);
      }
   }
}
```

5.1.2 GetValue

Description:

- **GetValue()**

Retrieve the amount of Value.

Parameters:

None, function takes no parameters.

Example

C#:

```
using System;
using  Algo.API;
using  Algo.API.Order;

{
   public  class  Sample: UserScript
   {
      public  Sample()
      {
         var account = Account.GetCurrentAccount();
         var value=account.GetValue();
         Print("Value"+value);
      }
   }
}
```

5.1.3 GetEquity

Description:

- **GetEquity()**

Retrieve the amount of Equity.

Parameters:

None, function takes no parameters.

Example

C#:

```
using System;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   public  class  Sample: UserScript
   {
      public  Sample()
      {
         var account = Account.GetCurrentAccount();
         var equity=account.GetEquity();
         Print("equity"+equity);
      }
   }
}
```

5.1.4 GetBuyingPower

Description:

- **GetBuying_power()**

Retrieve the amount of BuyingPower.

Parameters:

None, function takes no parameters.

Example

C#:

```
using System;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   public  class  Sample: UserScript
   {
      public  Sample()
      {
```

```
        var account = Account.GetCurrentAccount();
        var BuyingPower=account.GetBuying_Power ();
        Print("BuyingPower"+ BuyingPower);
    }
  }
}
```

5.1.5 GetLeverage_Ratio

Description:

- **GetLeverage_Ratio()**

Retrieve the valueof Leverage Ratio.

Parameters:

None, function takes no parameters.

Example

C#:

```
using System;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   public  class  Sample: UserScript
   {
      public  Sample()
      {
        var account = Account.GetCurrentAccount();
        var LeverageRatio =account.GetLeverage_Ratio();
        Print("Leverage Ratio"+ LeverageRatio);
      }
   }
}
```

5.1.6 GetRequiredMargin

Description:

- **GetRequiredMargin()**

Retrieve the value of RequiredMargin.

Parameters:

None, function takes no parameters.

Example

```
using System;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   public  class  Sample: UserScript
   {
      public  Sample()
      {
         var account = Account.GetCurrentAccount();
         var RequiredMargin =account.GetRequiredMargin();
         Print("RequiredMargin "+ RequiredMargin);
      }
   }
}
```

5.1.7 GetMaintenanceMargin

Description:

- **GetMaintenanceMargin ()**

Retrieve the value of MaintenanceMargin.

Parameters:

None, function takes no parameters.

Example

C#:

```
using System;
using  Algo.API;
using  Algo.API.Order;
namespace Test
```

```
{
   public  class  Sample: UserScript
   {
      public  Sample()
      {
         var account = Account.GetCurrentAccount();
         var MaintenanceMargin=account.GetMaintenanceMargin();
         Print("MaintenanceMargin"+ MaintenanceMargin);
      }
   }
}
```

5.1.8 GetLiquidationMargin

Description:

- **GetLiquidationMargin ()**

Retrieve the value of LiquidationMargin.

Parameters:

None, function takes no parameters.

Example

C#：

```
using System;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   public  class  Sample: UserScript
   {
      public  Sample()
      {
         var account = Account.GetCurrentAccount();
         var LiquidationMargin=account.GetLiquidationMargin();
         Print("LiquidationMargin"+ LiquidationMargin);
      }
   }
}
```

5.1.9 GetMargin_Ratio

Description:

- **GetMarginRatio ()**

Retrieve the value of MarginRatio.

Parameters:

None, function takes no parameters.

Example

C#：

```
using System;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   public  class  Sample: UserScript
   {
      public  Sample()
      {
         var account = Account.GetCurrentAccount();
         var MarginRatio=account.GetMargin_Ratio();
         Print("MarginRatio"+ MarginRatio);
      }
   }
}
```

5.1.10 GetCommission

Description:

- **GetCommission()**

Retrieve the value of Commission.

Parameters:

None, function takes no parameters.

Example

C#:

```
using System;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   public  class  Sample: UserScript
   {
      public  Sample()
      {
         var account = Account.GetCurrentAccount();
         var Commission=account.GetCommission();
         Print("Commission"+ Commission);
      }
   }
}
```

5.1.11 GetPosition

Description:

- **GetPosition ()**

Retrieve the amount of Position.

Parameters:

None, function takes no parameters.

Example

C#:

```
using System;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   public  class  Sample: UserScript
   {
      public  Sample()
      {
```

```
        var account = Account.GetCurrentAccount();
        var position=account.GetPosition();
        Print("position"+position);
    }
  }
}
```

5.1.12 GetOpenPL

Description:

- **GetOpenPL ()**

Retrieve the value of OpenPL.

Parameters:

None, function takes no parameters.

Example

C#:

```
using System;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   public  class  Sample: UserScript
   {
      public  Sample()
      {
        var account = Account.GetCurrentAccount();
       var OpenPL=account.GetOpenPL();
        Print("CurrentAccount Openpl is "+OpenPL);
      }
   }
}
```

5.1.13 getClosePL

Description:

- **getClosePL ()**

Retrieve the value of ClosePL.

Parameters:

None, function takes no parameters.

Example

C#:

```
using System;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   public  class  Sample: UserScript
   {
      public  Sample()
      {
         var account = Account.GetCurrentAccount();
         var ClosePL=account.GetClosePL();
         Print("CurrentAccount Closepl is "+ClosePL);
      }
   }
}
```

5.1.14 GetPL

Description:

- **GetPL ()**

Retrieve the value of PL.

Parameters:

None, function takes no parameters.

Example

C#:

```
using System;
using  Algo.API;
using  Algo.API.Order;
namespace Test
```

```
{
   public  class  Sample: UserScript
   {
      public  Sample()
      {
         var account = Account.GetCurrentAccount();
       var PL=account.GetPL();
         Print("CurrentAccount pl is "+PL);
            }
   }
}
```

5.1.15 GetQty

Description:

- **GetQty ()**

Retrieve the value of QTY.

Parameters:

None, function takes no parameters.

Example

C#:

```
using System;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   public  class  Sample: UserScript
   {
      public  Sample()
      {
         var account = Account.GetCurrentAccount();
          var Qty=account.GetQty();
         Print("CurrentAccount Qty is "+Qty);
            }
   }
}
```

5.1.16 GetTickets

Description:

- **GetTickets ()**

Retrieve the amount of Tickets.

Parameters:

None, function takes no parameters.

Example

C#：

```
using System;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   public  class  Sample: UserScript
   {
      public  Sample()
      {
         var account = Account.GetCurrentAccount();
       var Tickets=account.GetTickets();
         Print("CurrentAccount Tickets is "+Tickets);
            }
   }
}
```

5.1.17 GetAvailableMargin

Description:

- **GetAvailableMargin ()**

Retrieve the value of Available Margin.

Parameters:

None, function takes no parameters.

Example

C#:

```
using System;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   public  class  Sample: UserScript
   {
      public  Sample()
      {
         var account = Account.GetCurrentAccount();
      var AvailableMargin=account.GetAvailableMargin();
         Print("CurrentAccount AvailableMargin is "+AvailableMargin);
      }
   }
}
```

5.1.18 GetAssetBySymbolName

Description:

- **GetAssetBySymbolName** ([*symbolName*])

Retrieve the **Asset** object by symbol name.

Parameters:

1. *symbolName:* Optional. The symbol is the symbol of Portfolio window.

Example

C#:

```
using System;
using    Algo.API;
using    Algo.API.Order;
namespace Test
{
      public    class    Sample: UserScript
      {

      public override void OnInit()
```

```
            {
                API.EnableTrading();
            }
        public override void OnTick(int index)
            {
                var account = Account.GetCurrentAccount();
                var Asset=account.GetAssetBySymbolName ("EUR/USD");
                Print ("assert for Current account symbol EUR/USD is :" +
Asset.GetPosition ());
                var Asset1=account.GetAssetBySymbolName ("CHF/JPY");
                Print ("assert for Current account symbol CHF/JPY is :" +
Asset1.GetPosition ());
            }
        }
    }
```

5.2 Asset Object

Description:

Asset object includes some properties and methods about portfolio. There are two methods for obtaining asset object:

1. We can use the **getAssetBySymbolName** function to get Asset object, as follows:

```
var objAcc = new Account ("SA07");
var objAsset = objAcc.getAssetBySymbolName ("EUR/JPY");
```

2. We can use **new** method to create the Account object, as follows:

```
var objAsset = new Asset (AccountID, symbolName);
```

Note: AccountID is account name. This method can obtain any specified asset object. If the return value is 0 or a negative integer value in the Asset Object, it means the function fails to retrieve the quote.

Function List:

Asset object have the following functions:

NO.	Function	Description
1	**GetSymbol** ()	Retrieve the symbol
2	**GetPosition** ()	Retrieve the value of Position
3	**GetPrice** ()	Retrieve the Price
4	**GetOpenPL** ()	Retrieve the value of OpenPL
5	**GetClosePL** ()	Retrieve the value of ClosePL
6	**GetPL** ()	Retrieve the value of PL
7	**ClosePosition**(*trade*[, *quantity*] [,	Submit a close position operation

	expiry])	

5.2.1 GetSymbol

Description:

- **GetSymbol ()**

Retrieve the symbol in the portfolio window.

Parameters:

None, function takes no parameters.

Example

C#:

```
using System;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   public  class  Sample: UserScript
   {
      public override void OnInit()
                {
                    API.EnableTrading();
                }
           public override void OnTick(int index)
                {
                var account = Account.GetCurrentAccount();
                var portfolio=account.GetAssetBySymbolName ();
                var symbol= portfolio.GetSymbol();
                Print("CurrentAccount symbol name "+ symbol);
           }
   }
}
```

5.2.2 GetSide

Description:

- **GetSide()**

Retrieve the value of Side (Short or Long) in the portfolio window.

Parameters:

None, function takes no parameters.

Example

C#:

```
using System;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   public  class  Sample: UserScript
   {
      public override void OnInit()
                  {
                      API.EnableTrading();
                  }
            public override void OnTick(int index)
                  {
         var account = Account.GetCurrentAccount();
       var portfolio=account.GetAssetBySymbolName ();
         var Side = portfolio.GetSide();
         Print("CurrentAccountSymbol Side "+ Side);
      }
   }
}
```

5.2.3 GetPosition

Description:

- **GetPosition()**

Retrieve the value of Position in the portfolio window.

Parameters:

None, function takes no parameters.

Example

C#:

```
using System;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   public  class  Sample: UserScript
   {
      public override void OnInit()
                {
                    API.EnableTrading();
                }
           public override void OnTick(int index)
                {
         var account = Account.GetCurrentAccount();
      var portfolio=account.GetAssetBySymbolName ();
         var position= portfolio.GetPosition();
         Print("CurrentAccountSymbol position "+position);
      }
   }
}
```

5.2.4 GetPrice

Description:

- **GetPrice ()**

Retrieve the Price in the portfolio window.

Parameters:

None, function takes no parameters.

Example

C#:

```
using System;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   public  class  Sample: UserScript
```

```
  {
     public override void OnInit()
      {
         API.EnableTrading();
       }
       public override void OnTick(int index)
       {
       var account = Account.GetCurrentAccount();
       var portfolio=account.GetAssetBySymbolName ();
       var price= portfolio.GetPrice();
       Print("CurrentAccountSymbol price "+price);
       }
  }
}
```

5.2.5 GetOpenPL

Description:

- **GetOpenPL ()**

Retrieve the value of OpenPL in the portfolio window.

Parameters:

None, function takes no parameters.

Example

C#:

```
using System;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   public  class  Sample: UserScript
   {
      public override void OnInit()
       {
                    API.EnableTrading();
       }
       public override void OnTick(int index)
       {
       var account = Account.GetCurrentAccount();
```

```
        var portfolio=account.GetAssetBySymbolName ();
        var OpenPL=portfolio.GetOpenPL();
        Print("CurrentAccount Openpl is "+OpenPL);
        }
    }
}
```

5.2.6 GetClosePL

Description:

- **GetClosePL ()**

Retrieve the value of ClosePL in the portfolio window.

Parameters:

None, function takes no parameters.

Example

C#:

```
using System;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
    public  class  Sample: UserScript
    {
        public override void OnInit()
        {
            API.EnableTrading();
        }
        public override void OnTick(int index)
        {
            var account = Account.GetCurrentAccount();
```

```
        var portfolio=account.GetAssetBySymbolName ();
        var ClosePL= portfolio.GetClosePL();
        Print("CurrentAccountSymbol Closepl is "+ClosePL);
      }
    }
}
```

5.2.7 GetPL

Description:

- **GetPL ()**

Retrieve the value of PL in the portfolio window.

Parameters:

None, function takes no parameters.

Example

C#:

```
using System;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   public  class  Sample: UserScript
   {
      public override void OnInit()
      {
        API.EnableTrading();
      }
      public override void OnTick(int index)
      {
        var account = Account.GetCurrentAccount();
        var portfolio=account.GetAssetBySymbolName ();
        var PL= portfolio.GetPL();
        Print("CurrentAccountSymbol pl is "+PL);
      }
   }
}
```

5.2.8 closePosition

Description:

- **closePosition ()**

Submit a close position operation.

Parameters:

None, function takes no parameters.

Example

C#:

```
using System;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   public  class  Sample: UserScript
   {
          bool flag = false;
         public override void OnInit()
        {
          API.EnableTrading();

         }
         public override void OnTick(int index)
        {
            bool isHistory = API.IsHistoryBar();
          if(!isHistory && !flag)
          {
            var account = Account.GetCurrentAccount();
           var name=account.GetAccountName();
           var portfolio=account.GetAssetBySymbolName ("EUR/USD");
           var position= portfolio.GetPosition();
            Print("CurrentAccount position "+position);
           var closeposition= portfolio.ClosePosition () ;
                if((!string.IsNullOrEmpty(closeposition)))
          {
                Print ("should exec close position");
                Sleep(500);
```

```
                    position= portfolio.GetPosition();
                   Print("After close ,the Account position is "+position);
                    }
                flag= true;
              }
            }
          }
}
```

6. Order Manager Functions

6.1 Order Manager Object

Description:

OrderManager Object includes some properties and methods about order manager. It doesn't be created in your script.
If the return value is 0 or a negative integer value in the Order Manager Object, it means the function fails to retrieve the quote.

Function List:

OrderManager object have the following functions:

NO.	Function	Description
1	OrderManager.**getOrders**([*sAccountName*] [,*sSymbol*])	Retrieve the object of Order
2	OrderManager.**getOpenOrders**([*sAccountName*] [,*sSymbol*])	Retrieve the Open Orders
3	OrderManager.**getFilledOrders**([*sAccountName*] [,*sSymbol*])	Retrieve the Filled Orders
4	OrderManager.**getCancelledOrders**([*sAccountName*] [,*sSymbol*])	Retrieve the Cancelled Orders
5	OrderManager.**cancelOrder** (*sOrderID*)	Cancel the pending order
6	OrderManager.**teplaceOrder***(sOrder*, *dStopPrice*, *dLimitPrice*, *iNewQuantity*, *sNewOrderType*)	Replace the pending order

6.1.1 OrderManager.GetOrders

Description:

- **OrderManager.GetOrders (**[*sAccountName*],[*sSymbol*]**)**

Retrieve the object of Order.

Parameters:

- *sAccountName:* Optional. A String value.
- *sSymbol:* Optional. A String value.

Example

C#:

```
using System;
using System.Collections.Generic;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   public  class  Sample: UserScript
   {
      OrderManager manager = new OrderManager();
      public override void OnInit()
      {
         Print("Check account orders:");
         List<OrderInfo> list= manager.GetOrders("TEST01","EUR/USD");
         foreach(OrderInfo info in list)
         {
      Print(info.GetAccount().ToString());
      Print(info.OrderID);
      Print(info.GetSymbol().ToString());
      Print(info.GetQty().ToString());
      Print(info.GetFilledPrice().ToString());
      Print(info.GetStopPrice().ToString());
      Print(info.GetLimitPrice().ToString());
      Print(info.GetType().ToString());
      Print(info.GetMarketable().ToString());
      Print(info.GetStatus().ToString());
      Print(info.GetRoute().ToString());
         }
      }
   }
}
```

6.1.2 OrderManager.GetOpenOrders

Description:

OrderManager.GetOpenOrders ([*sAccountName*] [,*sSymbol*])

Retrieve the Open Orders.

Parameters:

- *sAccountName:* Optional. A String value.
- *sSymbol:* Optional. A String value.

Example

C#:

```
using System;
using System.Collections.Generic;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   public  class  Sample: UserScript
   {
      OrderManager manager = new OrderManager();
      public override void OnInit()
      {
         Print("Check account orders:");
         List<OrderInfo> list= manager.GetOpenOrders("FX01","EUR/USD");
         foreach(OrderInfo info in list)
         {
      Print(info.GetAccount().ToString());
      Print(info.OrderID);
      Print(info.GetSymbol().ToString());
      Print(info.GetQty().ToString());
      Print(info.GetFilledPrice().ToString());
      Print(info.GetStopPrice().ToString());
      Print(info.GetLimitPrice().ToString());
      Print(info.GetOrderType().ToString());
      Print(info.GetMarketable().ToString());
      Print(info.GetStatus().ToString());
      Print(info.GetRoute().ToString());
      Print(info.GetStop_Loss().ToString());
      Print(info.GetTake_Profit().ToString());
      Print(info.Order_GetTradeDate().ToString());
      Print(info.Order_GetTradeTime().ToString());
         }
      }
```

```
    }
}
```

6.1.3 OrderManager.GetFilledOrders

Description:

- **OrderManager. getFilledOrders** ([*sAccountName*],[*sSymbol*])

Retrieve the Filled Orders.

Parameters:

- *sAccountName:* Optional. A String value.
- *sSymbol:* Optional. A String value.

Example

C#:

```
using System;
using System.Collections.Generic;
using   Algo.API;
using   Algo.API.Order;
namespace Test
{
    public   class   Sample: UserScript
    {
        OrderManager manager = new OrderManager();
        public override void OnInit()
        {
            Print("Check account orders:");
            List<OrderInfo> list= manager.GetFilledOrders("","");
            foreach(OrderInfo info in list)
            {
                Print(info.GetAccount().ToString());
                Print(info.OrderID);
                Print(info.GetSymbol().ToString());
                Print(info.GetQty().ToString());
                Print(info.GetStopPrice().ToString());
                Print(info.GetLimitPrice().ToString());
                Print(info.GetOrderType().ToString());
                Print(info.GetMarketable().ToString());
                Print(info.GetStatus().ToString());
                Print(info.GetRoute().ToString());
```

```
                }
            }
        }
    }
```

6.1.4 OrderManager.GetCancelledOrders

Description:

- **OrderManager.GetCancelledOrders** ([*sAccountName*],[*sSymbol*])

Retrieve the Cancelled Orders.

Parameters:

- *sAccountName:* Optional. A String value.
- *sSymbol:* Optional. A String value.

Example

C#:

```
using System;
using System.Collections.Generic;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   public  class  Sample: UserScript
   {
      OrderManager manager = new OrderManager();
      public override void OnInit()
      {
         Print("Check account orders:");
         List<OrderInfo> list= manager.GetCancelledOrders("LISA","EUR/USD");
         foreach(OrderInfo info in list)
         {
      Print(info.GetAccount().ToString());
      Print(info.OrderID);
      Print(info.GetSymbol().ToString());
      Print(info.GetQty().ToString());
      Print(info.GetStopPrice().ToString());
      Print(info.GetLimitPrice().ToString());
      Print(info.GetOrderType().ToString());
```

```
        Print(info.GetMarketable().ToString());
        Print(info.GetStatus().ToString());
        Print(info.GetRoute().ToString());
            }
        }
    }
}
```

6.1.5 OrderManager.CancelOrder

Description:

- **OrderManager. CancelOrder (*sOrderID*)**

Cancel the pending order.

Parameters:

- *OrderID*: Required. The object of specified order.

Example

C#:

```
using System;
using System.Collections.Generic;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   public  class  Sample: UserScript
   {
      bool flag = false;
      public override void OnInit()
      {
        API.EnableTrading();
      }
      public override void OnTick(int index)
      {
          bool isHistory = API.IsHistoryBar();
        var limit_price = API.GetQuote ("ASK") - 0.02;
              if(!isHistory && !flag)
          {
            OrderRequest order = new OrderRequest();
            var Order1 =order.BuyLimit(10000,limit_price, "EUR/USD",0,0,"","GTC");
```

```
            flag= true;
        Print("Please to check the order1 enter success");
          API.Sleep(5000);
         OrderRequest request = new OrderRequest();
         request.CancelOrder(Order1);
        Print("Please to check the order1 cancel success");
        }
         }
    }
}
```

6.1.6 OrderManager.ReplaceOrder

Description:

- **OrderManager.ReplaceOrder** (*sOrder*, *dStopPrice*, *dLimitPrice*, *iNewQuantity*, *sNewOrderType*)

Replace the pending order.

Parameters:

- *sOrder*: Required. The ID of the specified order.
- *dStopPrice / dLimitPrice*: Required. See below : “newPrice1, newPrice2 Explain”.
- *iNewQuantity*: Required. A positive integer value.
- *sNewOrderType*: Required. A String value. The following constants are available:

Constant	Comment
market	Submit a market order.
limit	Submit a limit order.
stop	Submit a stop order.
threshold	Submit a threshold order.

NewPrice1, NewPrice2 Explain:

newOrderType	newPrice1	newPrice2
market	0	0
stop	Stop Price	0
limit	0	Limit Price
Threshold	LowerPrice	Upper Price

Example

C#:

```
using System;
```

```
using System.Collections.Generic;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   public  class  Sample:UserScript
   {
      bool IsOrderSended = false;
     bool flag = false;
      public override void OnInit()
      {
        API.EnableTrading();
      }
      public override void OnTick(int index)
      {
          bool isHistory = API.IsHistoryBar();
       double limit_price = API.GetQuote ("ASK") - 0.02;
              if(!isHistory &&! IsOrderSended)
          {
            OrderRequest order = new OrderRequest();
            var OrderID =order.BuyLimit(13000,limit_price,"EUR/USD",0,0,"","GTC");
            flag= true;
     Print("Please to check the order enter success");

     API.Sleep(5000);

     double replaceprice=API.GetQuote("ask","EUR/USD") - 0.05;


     OrderRequest request = new OrderRequest();
     request.ReplaceOrder(OrderID,0,replaceprice,11000,"Limit");
      IsOrderSended = true;
     flag = true;
     Print("Please to check the order1 replace success");
     }
      }
   }
}
```

6.2 Order Object

Description:

Order Object includes some properties and methods about order. There are two

methods for obtaining asset object:

(1) We can use the **GetOrders** function to get the order object, as follows:

```
var objOrder= OrderManager.getOrders ();
```

(2) We can use the **new** method to create the order object, as follows:

```
var objOrder = new Order (orderID);;
```

Note: ordered is the ID of the specified order。

If the return value is 0 or a negative integer value in the Order Object, it means the function fails to retrieve the quote.

Function List：

Order object have the following functions:

NO.	Function	Description
1	Order.**GetSymbol** ()	Retrieve the Symbol of pending order
2	Order.**getQty** ()	Retrieve the value of QTY
3	Order.**getPrice** ()	Retrieve the price of Order
4	Order.**getStopPrice** ()	Retrieve the StopPrice of Order
5	Order.**getLimitPrice** ()	Retrieve the LimitPrice of Order
6	Order.**getType** ()	Retrieve the orderType of Order
7	Order.**getMarketable** ()	Retrieve the properties of Order
8	Order.**getStatus** ()	Retrieve the status of Order
9	Order.**getRoute** ()	Retrieve the Route of Order
10	Order.**cancel** ()	Submit a cancel order operation

6.2.1 Order.GetSymbol

Description:

- **Order.GetSymbol ()**

Retrieve the Symbol of pending order.

Parameters:

None, function takes no parameters.

Example

C#：

```
using System;
using System.Collections.Generic;
using  Algo.API;
```

```
using  Algo.API.Order;
namespace Test
{
   public  class  Sample: UserScript
   {
       OrderInfo info =new OrderInfo("14574");
       public override void OnInit()
       {
      Print(info.GetSymbol().ToString());
       }
    }
 }
```

6.2.2 Order.GetQty

Description:

- **Order.GetQty ()**

Retrieve the value of QTY.

Parameters:

None, function takes no parameters.

Example

C#:

```
using System;
using System.Collections.Generic;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   public  class  Sample: UserScript
   {
       OrderInfo info =new OrderInfo("14574");
       public override void OnInit()
       {
      Print(info.GetQty().ToString());
      }
      }
      }
```

6.2.3 Order.GetFilledPrice

Description:

- **Order.GetFilledPrice ()**

Retrieve the price of Order.

Parameters:

None, function takes no parameters.

Example

C#:

```
using System;
using System.Collections.Generic;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   public  class  Sample: UserScript
   {
      OrderInfo info =new OrderInfo("14574");
      public override void OnInit()
      {
     Print(info.GetFilledPrice().ToString());
     }
     }
     }
```

6.2.4 Order.GetStopPrice

Description:

- **Order. GetStopPrice ()**

Retrieve the StopPrice of Order.

Parameters:

None, function takes no parameters.

Example

C#:

```
using System;
using System.Collections.Generic;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   public  class  Sample: UserScript
   {
      OrderInfo info =new OrderInfo("14574");
      public override void OnInit()
      {
     Print(info.GetStopPrice().ToString());
     }
     }
     }
```

6.2.5 Order.GetLimitPrice

Description:

- **Order. GetLimitPrice ()**

Retrieve the LimitPrice of Order.

Parameters:

None, function takes no parameters.

Example

C#:

```
using System;
using System.Collections.Generic;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   public  class  Sample: UserScript
```

```
{
    OrderInfo info =new OrderInfo("14574");
    public override void OnInit()
    {
   Print(info.GetLimitPrice().ToString());
   }
   }
   }
```

6.2.6 Order.GetType

Description:

- **Order. GetType ()**

Retrieve the orderType of Order.

Parameters:

None, function takes no parameters.

Example

C#:

```
using System;
using System.Collections.Generic;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   public  class  Sample: UserScript
   {
       OrderInfo info =new OrderInfo("14574");
       public override void OnInit()
       {
      Print(info.GetOrderType().ToString());
      }
      }
      }
```

6.2.7 Order.GetMarketable

Description:

- **Order. GetMarketable ()**

Retrieve the properties of Order.

Parameters:

None, function takes no parameters.

Example

C#:

```
using System;
using System.Collections.Generic;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   public  class  Sample: UserScript
   {
      OrderInfo info =new OrderInfo("14574");
      public override void OnInit()
      {
      Print(info.GetMarketable().ToString());
      }
      }
      }
```

6.2.8 Order.GetStatus

Description:

- **Order. GetStatus ()**

Retrieve the status of Order.

Parameters:

None, function takes no parameters.

Example

C#:

```
using System;
```

```
using System.Collections.Generic;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   public  class  Sample: UserScript
   {
      OrderInfo info =new OrderInfo("14574");
      public override void OnInit()
      {
     Print(info.GetStatus().ToString());
     }
     }
     }
```

6.2.9 Order.GetRoute

Description:

- **Order. GetRoute ()**

Retrieve the Route of Order.

Parameters:

None, function takes no parameters.

Example

C#:

```
     using System;
using System.Collections.Generic;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   public  class  Sample: UserScript
   {
      OrderInfo info =new OrderInfo("14574");
      public override void OnInit()
      {
     Print(info.GetRoute().ToString());
```

```
}
    }
    }
```

6.2.10 Order.Getside

Description:

- **Order. GetSide ()**

Retrieve the Side of Order.

Parameters:

None, function takes no parameters.

Example

C#：

```
using System;
using System.Collections.Generic;
using Algo.API;
using Algo.API.Order;
namespace Test
{
public class Sample:UserScript
{
OrderInfo info =new OrderInfo("14574");
public override void OnInit()
{
Print(info.GetSide().ToString());
}
}
}
```

6.2.11 Order.GetTradeDate

Description:

- **Order. GetTradeDate()**

Retrieve the TradeDate of Order.

Parameters:

None, function takes no parameters.

Example

C#:

```
using System;
using System.Collections.Generic;
using Algo.API;
using Algo.API.Order;
namespace Test
{
public class Sample:UserScript
{
OrderInfo info =new OrderInfo("14574");
public override void OnInit()
{
Print(info.GetTradeDate ().ToString());
}
}
}
```

6.2.12 Order.GetTradeTime

Description:

- **Order. GetTradeTime()**

Retrieve the TradeTime of Order.

Parameters:

None, function takes no parameters.

Example

C#:

```
using System;
using System.Collections.Generic;
using Algo.API;
using Algo.API.Order;
namespace Test
```

```
    {
    public class Sample:UserScript
    {
    OrderInfo info =new OrderInfo("14574");
    public override void OnInit()
    {
    Print(info.GetTradeTime().ToString());
    }
    }
}
```

6.2.13 Order.GetStop_Loss

Description:

- **Order.GetStop_Loss()**

Retrieve the StopLoss Price of Order.

Parameters:

None, function takes no parameters.

Example

C#:

```
    using System;
    using System.Collections.Generic;
    using Algo.API;
    using Algo.API.Order;
    namespace Test
    {
    public class Sample:UserScript
    {
    OrderInfo info =new OrderInfo("14574");
    public override void OnInit()
    {
    Print(info.GetStop_Loss().ToString());
    }
    }
    }
```

6.2.14 Order.GetTake_Profit

Description:

- **Order.GetTake_Profit()**

Retrieve the Take_Profit Price of Order.

Parameters:

None, function takes no parameters.

Example

C#:

```
using System;
using System.Collections.Generic;
using Algo.API;
using Algo.API.Order;
namespace Test
{
public class Sample:UserScript
{
OrderInfo info =new OrderInfo("14574");
public override void OnInit()
{
Print(info.GetTake_Profit().ToString());
}
}
}
```

6.2.15 Order.cancel

Description:

- **Order. cancel ()**

Submit a cancel order operation.

Parameters:

None, function takes no parameters.

Example

C#

```
using System;
using System.Collections.Generic;
using   Algo.API;
using   Algo.API.Order;
namespace Test
{
      public   class   Sample:UserScript
      {
            bool flag = false;
            public override void OnInit()
            {
                  API.EnableTrading();
            }
            public override void OnTick(int index)
            {
                  bool isHistory = API.IsHistoryBar();
      var limit_price = API.GetQuote ("ASK") - 0.02;
                  if(!isHistory && !flag)
                  {
                      OrderRequest order = new OrderRequest();
                      var Order1 =order.BuyLimit(10000,limit_price,
"EUR/USD",0,0,"","GTC");
                      flag= true;
                      Print("Please to check the order1 enter success");
                        API.Sleep(5000);
                      OrderInfo info =new OrderInfo(Order1);
                      info.Cancel();
                      Print("Please to check the order1 cancel success");
                      }
            }
      }
}
```

7. Utility Functions

Comment：No such Functions in C#.

7.1 FunctionParameter Object

Description:

- **FunctionParameter (***paramString***, *ParamType*)**

FunctionParameter Object class is a set of objects and functions that allow you to manage your input parameters in a script.

Parameters:

- *paramString*: Required. the name of the parameter as defined in main()
- *paramType*: Required. The type of parameter object. see Parameter Types below. The following constants are available:

Constant	Comment
FunctionParameter.**STRING**	A string menu object. either a single string or a list of string options
FunctionParameter.**NUMBER**	A numeric menu object
FunctionParameter.**COLOR**	A color menu object
FunctionParameter.**BOOLEAN**	A Boolean menu object

FunctionParameter Methods:

Method	Comment
setName()	Set the display name of this menu item. This allows you to create a more descriptive name to display to the user
addOption()	Add a string option
setLowerLimit()	Set the lowest acceptable input value for a numeric menu option
setUpperLimit()	Set the highest acceptable input value for a numeric menu option
setDefault()	Set the default value for this menu option

Example

7.2 call

Description:

- **call (***efsname* [,*arguments*])**

This function is provided for backwards compatibility with EFS1. It allows you to call another script and retrieve the return values from that script.

Parameters:

- *efsname*: Required. A String value. The path and filename of the script to be called.
- *arguments:* Optional. A String value. list of input parameters, if any, required by the script being called

Example

7.3 internalCall

Description:

internalcall (*functionName* [, *parameters*...]**)**

Returns a series object representing the values calculated from a given function name. The function specified will be executed as if it were a Main() function being called in an external EFS.

Parameters:

- *functionName*: Required. A String value. name of user-defined function to call.
- *parameters:* Optional. A String value. Parameters, if any, required by the user-defined function you are calling.

Note:

If you pass a series object to the **internalCalll** () function, be sure to pass the series as the last parameter. This will force **internalCall** () to execute in the series' sym/inv context.

Example

7.4 ref

Description:

ref (*barIndex***)**

The **ref**() function allows you to retrieve prior values for a built-in study (or studies). If multiple return values are being used then the **ref**() function will return its results in an array.

Parameters:

- *barIndex*: Required. offset indicating which historical indicator data points to retrieve

Example

8. Quote Object

Description:

The **Quote** Object class is a set of field that allows you to store the quote information in script.

8.1 Quote info

we can use getQuoteInfo function to get the last valid Quote object:

var objOrder= getQuote();

Attribute List:

The object contains 15 field values, show below:

NO.	Attribute	Comment
1	**Symbol**	Retrieve the name of the symbol currently being charted
2	**Bid**	Retrieve the bid price
3	**Ask**	Retrieve the ask price
4	**BidSize**	Retrieve current symbol's bid size
5	**AskSize**	Retrieve current symbol's ask size
6	**BidMMID**	bidExchange of L1
7	**AskMMID**	askExchange of L1
8	**Spread**	Retrieve the value of spread
9	**QuoteTick**	Tick for quote price

Example:

C#:

```
using System;
using  Algo.API;
using  Algo.API.Order;
namespace Test
{
   public  class  Sample: UserScript
   {
      public override void OnInit()
```

```
    {
      API.EnableTrading();
    }
     Quote quote;
    public override void OnTick(int index)
    {
   if(API.IsHistoryBar())
    {
        return;
    }
        quote =Quote.GetQuote();
        Print ("sym:" + quote.Symbol);
        Print ("bid:" + quote.Bid);
        Print ("ask:" + quote.Ask);
        Print ("bidsize:" + quote.BidSize);
        Print ("asksize:" + quote.AskSize);
        Print ("bidmmid:" + quote.BidMMID);
        Print ("askmmid:" + quote.AskMMID);
        Print ("spread:" + quote.Spread);
        Print ("quotetick:" + quote.QuoteTick);
    }
  }
}
```

9. Appendix

Comment：No such Functions in C#.

9.1 Available Colors

Constant	Comment
Color.**white**	Sample
Color.**black**	Sample
Color.**darkgrey**	Sample
Color.**grey**	Sample
Color.**lightgrey**	Sample
Color.**navy**	Sample
Color.**blue**	Sample
Color.**aqua**	Sample
Color.**cyan**	Sample
Color.**teal**	Sample
Color.**darkgreen**	Sample
Color.**green**	Sample

Color.**lime**	Sample
Color.**olive**	Sample
Color.**khaki**	Sample
Color.**brown**	Sample
Color.**purple**	Sample
Color.**red**	Sample
Color.**magenta**	Sample
Color.**maroon**	Sample
Color.**yellow**	Sample
Color.**lightyellow**	Sample
Color.**paleyellow**	Sample