

```
/******
```

This file contains the `scsf_CopyOfBaseGraph` function which is an Advanced Custom Study function that creates a copy of the existing base graph in a chart. You can modify this or use it as an example to create your own custom chart.

If you want to create your own custom chart bars which are based upon the underlying data in an Intraday chart data file, instead used the custom chart bar interface which is demonstrated through the included `ACSILCustomChartBars_Example.cpp` file.

`scsf_CopyOfBaseGraph` is used to create custom chart bars but are based upon the existing chart bars loaded within the chart. This may be useful for some purposes, but in general it is an inefficient method to create custom chart bars.

```
*****/
```

```
#include "sierrachart.h"
```

```
/*=====*/
```

```
SCSFExport scsf_CopyOfBaseGraph(SCStudyInterfaceRef sc)
```

```
{
    SCSubgraphRef Subgraph_Open = sc.Subgraph[0];
    SCSubgraphRef Subgraph_High = sc.Subgraph[1];
    SCSubgraphRef Subgraph_Low = sc.Subgraph[2];
    SCSubgraphRef Subgraph_Last = sc.Subgraph[3];
    SCSubgraphRef Subgraph_Volume = sc.Subgraph[4];
    SCSubgraphRef Subgraph_OpenInterest = sc.Subgraph[5];
    SCSubgraphRef Subgraph_OHLCAvg = sc.Subgraph[6];
    SCSubgraphRef Subgraph_HLCAvg = sc.Subgraph[7];
    SCSubgraphRef Subgraph_HLAvg = sc.Subgraph[8];
    //SCSubgraphRef BidVol = sc.Subgraph[9];
    // SCSubgraphRef AskVol = sc.Subgraph[10];

    // Set configuration variables

    if (sc.SetDefaults)
    {
        // Set the defaults

        sc.GraphName = "Copy of BaseGraph Example";

        sc.StudyDescription = "This is an example of how to create a custom chart. This one is a copy of the BaseGraph.";

        // This member indicates that this is a custom chart study.
        // This can only be set when SetDefaults is true.
        sc.IsCustomChart = 1; // true
        sc.GraphRegion = 0; // First region
        sc.DrawZeros = 0; // false
        sc.GraphDrawType = GDT_OHLCBAR;
        sc.StandardChartHeader = 1; // true

        // Subgraphs

        Subgraph_Open.Name = "Open";
        Subgraph_Open.DrawStyle = DRAWSTYLE_LINE;

        Subgraph_High.Name = "High";
        Subgraph_High.DrawStyle = DRAWSTYLE_LINE;

        Subgraph_Low.Name = "Low";
        Subgraph_Low.DrawStyle = DRAWSTYLE_LINE;
```

```

Subgraph_Last.Name = "Last";
Subgraph_Last.DrawStyle = DRAWSTYLE_LINE;

Subgraph_Volume.Name = "Volume";
Subgraph_Volume.DrawStyle = DRAWSTYLE_IGNORE;

Subgraph_OpenInterest.Name = "# of Trades / OI";
Subgraph_OpenInterest.DrawStyle = DRAWSTYLE_IGNORE;

Subgraph_OHLCAvg.Name = "OHLC Avg";
Subgraph_OHLCAvg.DrawStyle = DRAWSTYLE_IGNORE;

Subgraph_HLCAvg.Name = "HLC Avg";
Subgraph_HLCAvg.DrawStyle = DRAWSTYLE_IGNORE;

Subgraph_HLAvg.Name = "HL Avg";
Subgraph_HLAvg.DrawStyle = DRAWSTYLE_IGNORE;

/* BidVol.Name = "Bid Vol";
BidVol.DrawStyle = DRAWSTYLE_IGNORE;

AskVol.Name = "Ask Vol";
AskVol.DrawStyle = DRAWSTYLE_IGNORE;*/

return;
}

// If our start index is zero, then we want to clear our existing chart
// data, if any, so we start clean to avoid any possible problems. This
// is accomplished by using sc.ResizeArrays.
if (sc.UpdateStartIndex == 0)
{
    sc.ResizeArrays(0);
}

// Do data processing
for (int i = sc.UpdateStartIndex; i < sc.ArraySize; i++)
{
    if (sc.OutArraySize - 1 < i)
        sc.AddElements(1);

    sc.DateTimeOut[i] = sc.BaseDateIn[i];

    for (int SubGraph = 0; SubGraph <= 8; SubGraph++)
    {
        sc.Subgraph[SubGraph][i] = sc.BaseDataIn[SubGraph][i];

        //For testing coloring bars
        //sc.Subgraph[SubGraph].DataColor[i] = RGB(255,255,255);
    }
}
}

/*=====*/
SCSFExport scsf_ReverseOrderCopyOfBaseGraph(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_Open = sc.Subgraph[0];
    SCSubgraphRef Subgraph_High = sc.Subgraph[1];
    SCSubgraphRef Subgraph_Low = sc.Subgraph[2];
    SCSubgraphRef Subgraph_Last = sc.Subgraph[3];
    SCSubgraphRef Subgraph_Volume = sc.Subgraph[4];
    SCSubgraphRef Subgraph_OpenInterest = sc.Subgraph[5];
    SCSubgraphRef Subgraph_OHLCAvg = sc.Subgraph[6];
    SCSubgraphRef Subgraph_HLCAvg = sc.Subgraph[7];

```

```

SCSubgraphRef Subgraph_HLAvg = sc.Subgraph[8];
SCSubgraphRef Subgraph_BidVol = sc.Subgraph[9];
SCSubgraphRef Subgraph_AskVol = sc.Subgraph[10];

SCInputRef Input_InNumberOfBarsToReverse = sc.Input[1];

// Set configuration variables

if (sc.SetDefaults)
{
    // Set the defaults
    sc.GraphName = "Reverse Copy of Base Graph";

    sc.AutoLoop = 0;

    // This member indicates that this is a custom chart study.
    // This can only be set when SetDefaults is true.
    sc.IsCustomChart = 1; // true
    sc.GraphRegion = 0; // First region
    sc.DrawZeros = 0; // false
    sc.GraphDrawType = GDT_OHLCBAR;
    sc.StandardChartHeader = 1; // true
    sc.GraphUsesChartColors = 1;

    Input_InNumberOfBarsToReverse.Name = "Number Of Bars To Reverse";
    Input_InNumberOfBarsToReverse.SetInt(1000);
    Input_InNumberOfBarsToReverse.SetIntLimits(10, 2000);

    return;
}

int &PriorBaseGraphArraySize = sc.GetPersistentInt(1);

// If there is a full recalculation, then we want to clear the existing chart
// data, if any, so we start clean to avoid any possible problems. This
// is accomplished by using sc.ResizeArrays.
if (sc.IsFullRecalculation && sc.UpdateStartIndex==0)
{
    sc.ResizeArrays(0);

    for (int SubgraphIndex = 0; SubgraphIndex <= NUM_BASE_GRAPH_ARRAYS; ++SubgraphIndex)
    {
        sc.Subgraph[SubgraphIndex].Name = sc.GetStudySubgraphName(0, SubgraphIndex);
    }

    PriorBaseGraphArraySize = 0;
}

int RequiredBarElements = min(sc.ArraySize, Input_InNumberOfBarsToReverse.GetInt());
if (sc.OutArraySize < RequiredBarElements)
    sc.ResizeArrays(RequiredBarElements);
//sc.AddElements(1);

// Do data processing
for (int InputDataIndex = sc.ArraySize - 1, OutputIndex = 0; InputDataIndex >= 0; InputDataIndex--, OutputIndex++)
{
    sc.DateTimeOut[OutputIndex] = sc.BaseDateTimeln[InputDataIndex];

    for (int SubGraph = 0; SubGraph <= NUM_BASE_GRAPH_ARRAYS; SubGraph++)
    {
        sc.Subgraph[SubGraph][OutputIndex] = sc.BaseDataIn[SubGraph][InputDataIndex];
    }

    //if there is no change in based graph array size, then one iteration is sufficient

```

```
    if (PriorBaseGraphArraySize == sc.ArraySize)
        break;
}

PriorBaseGraphArraySize = sc.ArraySize;
}
```