

```
#ifndef _SCCONSTANTS_H_
#define _SCCONSTANTS_H_
```

//Refer to ACSIL Arrays and Variables documentation for descriptions of each of these sc.BaseDataIn[] constants

```
const int SC_OPEN      = 0;
const int SC_HIGH      = 1;
const int SC_LOW       = 2;
const int SC_LAST      = 3;
const int SC_VOLUME    = 4;
const int SC_OPEN_INTEREST = 5;
const int SC_NUM_TRADES = 5;
const int SC_OHLC_AVG  = 6;
const int SC_OHLC      = 6;
const int SC_HLC_AVG   = 7;
const int SC_HLC       = 7;
const int SC_HL_AVG    = 8;
const int SC_HL        = 8;
const int SC_BIDVOL    = 9;
const int SC_ASKVOL    = 10;
const int SC_UPVOL     = 11;
const int SC_DOWNVOL   = 12;
const int SC_BIDNT     = 13;
const int SC_ASKNT     = 14;
const int SC_ASKBID_DIFF_HIGH = 15;
const int SC_ASKBID_DIFF_LOW  = 16;
const int SC_ASKBID_NUM_TRADES_DIFF_HIGH = 17; //SC_ASKBID_NT_DIFF_HIGH
const int SC_ASKBID_NUM_TRADES_DIFF_LOW  = 18; //SC_ASKBID_NT_DIFF_LOW

const int SC_UPDOWN_VOL_DIFF_HIGH = 19;
const int SC_UPDOWN_VOL_DIFF_LOW  = 20;

const int SC_RENKO_OPEN      = 21;
const int SC_POINT_FIGURE_HIGH = SC_RENKO_OPEN;
const int SC_RENKO_CLOSE     = 22;
const int SC_POINT_FIGURE_LOW = SC_RENKO_CLOSE;

const int SC_BID_PRICE = 23;
const int SC_ASK_PRICE = 24;
const int SC_ASK_BID_VOL_DIFF_MOST_RECENT_CHANGE = 25;

const int NUM_BASE_GRAPH_ARRAYS = SC_ASKVOL + 1;
const int NUM_BASE_GRAPH_ARRAYS_WITH_ADDITIONAL_ARRAYS =
SC_ASK_BID_VOL_DIFF_MOST_RECENT_CHANGE + 1;

const int NUM_BASE_GRAPHS_FOR_HISTORICAL_CHART = SC_ASKVOL + 1;

const int NUM_BASE_GRAPH_ARRAYS_WITH_RENKO_ARRAYS = SC_RENKO_CLOSE + 1;

const int PF_NUM_BOXES_ARRAY = NUM_BASE_GRAPH_ARRAYS_WITH_ADDITIONAL_ARRAYS + 0;
const int PF_DIRECTION_ARRAY = NUM_BASE_GRAPH_ARRAYS_WITH_ADDITIONAL_ARRAYS + 1;

const int NUM_BASE_GRAPH_ARRAYS_WITH_POINT_FIGURE_ARRAYS = PF_DIRECTION_ARRAY + 1;

const int PF_TRUELAST_ARRAY = NUM_BASE_GRAPH_ARRAYS_WITH_ADDITIONAL_ARRAYS + 2; //P&F study
only

const int THREE_LINE_BREAK_LBOPEN_ARRAY = NUM_BASE_GRAPH_ARRAYS_WITH_ADDITIONAL_ARRAYS +
0;
const int THREE_LINE_BREAK_REVERSE_ARRAY = NUM_BASE_GRAPH_ARRAYS_WITH_ADDITIONAL_ARRAYS
+ 1;
const int THREE_LINE_BREAK_CONTINUE_ARRAY = NUM_BASE_GRAPH_ARRAYS_WITH_ADDITIONAL_ARRAYS
+ 2;

const int SC_NO = 0;
const int SC_YES = 1;
```

```
const int MAX_STUDY_LENGTH = 1000000;
```

```
// Used in s_ChartDrawing
```

```
// These are written to a file
```

```
enum DrawingTypeEnum
```

```
{ DRAWING_UNKNOWN = 0
```

```
, DRAWING_LINE = 1
```

```
, DRAWING_RAY = 2
```

```
, DRAWING_HORIZONTALLINE = 3
```

```
, DRAWING_VERTICALLINE = 4
```

```
, DRAWING_ARROW = 5
```

```
, DRAWING_TEXT = 6
```

```
, DRAWING_CALCULATOR = 7
```

```
, DRAWING_RETRACEMENT = 8
```

```
, DRAWING_UNUSED = 9 // old fan drawing type
```

```
, DRAWING_UNUSED_10 = 10
```

```
, DRAWING_PRICE_PROJECTION = 11
```

```
, DRAWING_RECTANGLEHIGHLIGHT = 12
```

```
, DRAWING_ELLIPSEHIGHLIGHT = 13
```

```
, DRAWING_FAN_GANN = 14
```

```
, DRAWING_PITCHFORK = 15
```

```
, DRAWING_UNUSED16 = 16 // was DRAWING_FLOWELLIPSEHIGHLIGHT
```

```
, DRAWING_WORKING_ORDER = 17
```

```
, DRAWING_POSITION_LINE = 18
```

```
, DRAWING_CYCLE = 19
```

```
, DRAWING_TIME_EXPANSION = 20
```

```
, DRAWING_GANNGRID = 21
```

```
, DRAWING_UNUSED_22 = 22
```

```
, DRAWING_UNUSED_23 = 23
```

```
, DRAWING_ORDER_FILL = 24
```

```
, DRAWING_ENTRYEXIT_CONNECTLINE = 25
```

```
, DRAWING_RECTANGLE_EXT_HIGHLIGHT = 26
```

```
, DRAWING_FAN_FIBONACCI = 27
```

```
, DRAWING_PARALLEL_LINES = 28
```

```
, DRAWING_PARALLEL_RAYS = 29
```

```
, DRAWING_LINEAR_REGRESSION = 30
```

```
, DRAWING_RAFF_REGRESSION_CHANNEL = 31
```

```
, DRAWING_EXTENDED_LINE = 32
```

```
, DRAWING_PITCHFORK_SCHIFF = 33
```

```
, DRAWING_PITCHFORK_MODIFIED_SCHIFF = 34
```

```
, DRAWING_PRICE_EXPANSION = 35
```

```
, DRAWING_VOLUME_PROFILE = 36
```

```
, DRAWING_STATIONARY_TEXT = 37
```

```
, DRAWING_TIME_PROJECTION = 38
```

```
, DRAWING_MARKER = 39
```

```
, DRAWING_HORIZONTAL_RAY = 40
```

```
, DRAWING_REWARD_RISK = 41
```

```
, DRAWING_SWING_MARKER = 42
```

```
, DRAWING_DATE_MARKER = 43
```

```
, DRAWING_OHLC_RAY = 44
```

```
, DRAWING_HORIZONTAL_LINE_NON_EXTENDED = 45
```

```
, DRAWING_TRIANGLE = 46
```

```
, DRAWING_ANGLED_ELLIPSE = 47
```

```
, SC_NUM_DRAWING_TYPE
```

```
};
```

```
const double CHART_DRAWING_MAX_HORIZONTAL_AXIS_RELATIVE_POSITION = 150.0;
```

```
const double CHART_DRAWING_MAX_VERTICAL_AXIS_RELATIVE_POSITION = 100.0;
```

```
const double CHART_DRAWING_MAX_HORIZONTAL_AXIS_RELATIVE_POSITION_HIGH_RES = 10000.0;
```

```
const double CHART_DRAWING_MAX_VERTICAL_AXIS_RELATIVE_POSITION_HIGH_RES = 10000.0;
```

```
//For deleting Chart Drawings
```

```
const int TOOL_DELETE_ALL          = -1;
const int TOOL_DELETE_CHARTDRAWING = 0;
```

```
// Marker Chart Drawing constants
```

```
const int MARKER_POINT      = 0;
const int MARKER_DASH       = 1;
const int MARKER_SQUARE     = 2;
const int MARKER_STAR       = 3;
const int MARKER_PLUS       = 4;
const int MARKER_X          = 5;
const int MARKER_ARROWUP    = 6;
const int MARKER_ARROWDOWN  = 7;
const int MARKER_ARROWRIGHT = 8;
const int MARKER_ARROWLEFT  = 9;
const int MARKER_TRIANGLEUP = 10;
const int MARKER_TRIANGLEDOWN = 11;
const int MARKER_TRIanglerIGHT = 12;
const int MARKER_TRIANGLELEFT = 13;
const int MARKER_DIAMOND    = 14;
```

```
const int TIME_EXP_LABEL_NONE      = 0;
const int TIME_EXP_LABEL_PERCENT   = 1;
const int TIME_EXP_LABEL_BARS      = 2;
const int TIME_EXP_LABEL_PERCENTBARS = 3;
const int TIME_EXP_LABEL_BARSPERCENT = 4;
const int TIME_EXP_LABEL_DATE      = 5;
const int TIME_EXP_LABEL_TIME      = 6;
const int TIME_EXP_LABEL_DATETIME  = 7;
```

```
const int TIME_EXP_EXTENDED = 0;
const int TIME_EXP_STANDARD = 1;
const int TIME_EXP_COMPRESSED = 2;
```

```
// Constants for ACS Control Bar buttons
```

```
const int ACS_BUTTON_1 = 1;
const int ACS_BUTTON_2 = 2;
const int ACS_BUTTON_3 = 3;
const int ACS_BUTTON_4 = 4;
const int ACS_BUTTON_5 = 5;
const int ACS_BUTTON_6 = 6;
const int ACS_BUTTON_7 = 7;
const int ACS_BUTTON_8 = 8;
const int ACS_BUTTON_9 = 9;
const int ACS_BUTTON_10 = 10;
const int ACS_BUTTON_11 = 11;
const int ACS_BUTTON_12 = 12;
const int ACS_BUTTON_13 = 13;
const int ACS_BUTTON_14 = 14;
const int ACS_BUTTON_15 = 15;
const int ACS_BUTTON_16 = 16;
const int ACS_BUTTON_17 = 17;
const int ACS_BUTTON_18 = 18;
const int ACS_BUTTON_19 = 19;
const int ACS_BUTTON_20 = 20;
const int ACS_BUTTON_21 = 21;
const int ACS_BUTTON_22 = 22;
const int ACS_BUTTON_23 = 23;
const int ACS_BUTTON_24 = 24;
const int ACS_BUTTON_25 = 25;
const int ACS_BUTTON_26 = 26;
const int ACS_BUTTON_27 = 27;
const int ACS_BUTTON_28 = 28;
const int ACS_BUTTON_29 = 29;
const int ACS_BUTTON_30 = 30;
const int ACS_BUTTON_31 = 31;
```

```
const int ACS_BUTTON_32 = 32;
const int ACS_BUTTON_33 = 33;
const int ACS_BUTTON_34 = 34;
const int ACS_BUTTON_35 = 35;
const int ACS_BUTTON_36 = 36;
const int ACS_BUTTON_37 = 37;
const int ACS_BUTTON_38 = 38;
const int ACS_BUTTON_39 = 39;
const int ACS_BUTTON_40 = 40;
const int ACS_BUTTON_41 = 41;
const int ACS_BUTTON_42 = 42;
const int ACS_BUTTON_43 = 43;
const int ACS_BUTTON_44 = 44;
const int ACS_BUTTON_45 = 45;
const int ACS_BUTTON_46 = 46;
const int ACS_BUTTON_47 = 47;
const int ACS_BUTTON_48 = 48;
const int ACS_BUTTON_49 = 49;
const int ACS_BUTTON_50 = 50;
const int ACS_BUTTON_51 = 51;
const int ACS_BUTTON_52 = 52;
const int ACS_BUTTON_53 = 53;
const int ACS_BUTTON_54 = 54;
const int ACS_BUTTON_55 = 55;
const int ACS_BUTTON_56 = 56;
const int ACS_BUTTON_57 = 57;
const int ACS_BUTTON_58 = 58;
const int ACS_BUTTON_59 = 59;
const int ACS_BUTTON_60 = 60;
const int ACS_BUTTON_61 = 61;
const int ACS_BUTTON_62 = 62;
const int ACS_BUTTON_63 = 63;
const int ACS_BUTTON_64 = 64;
const int ACS_BUTTON_65 = 65;
const int ACS_BUTTON_66 = 66;
const int ACS_BUTTON_67 = 67;
const int ACS_BUTTON_68 = 68;
const int ACS_BUTTON_69 = 69;
const int ACS_BUTTON_70 = 70;
const int ACS_BUTTON_71 = 71;
const int ACS_BUTTON_72 = 72;
const int ACS_BUTTON_73 = 73;
const int ACS_BUTTON_74 = 74;
const int ACS_BUTTON_75 = 75;
const int ACS_BUTTON_76 = 76;
const int ACS_BUTTON_77 = 77;
const int ACS_BUTTON_78 = 78;
const int ACS_BUTTON_79 = 79;
const int ACS_BUTTON_80 = 80;
const int ACS_BUTTON_81 = 81;
const int ACS_BUTTON_82 = 82;
const int ACS_BUTTON_83 = 83;
const int ACS_BUTTON_84 = 84;
const int ACS_BUTTON_85 = 85;
const int ACS_BUTTON_86 = 86;
const int ACS_BUTTON_87 = 87;
const int ACS_BUTTON_88 = 88;
const int ACS_BUTTON_89 = 89;
const int ACS_BUTTON_90 = 90;
const int ACS_BUTTON_91 = 91;
const int ACS_BUTTON_92 = 92;
const int ACS_BUTTON_93 = 93;
const int ACS_BUTTON_94 = 94;
const int ACS_BUTTON_95 = 95;
const int ACS_BUTTON_96 = 96;
```

```
const int ACS_BUTTON_97 = 97;
const int ACS_BUTTON_98 = 98;
const int ACS_BUTTON_99 = 99;
const int ACS_BUTTON_100 = 100;
const int ACS_BUTTON_101 = 101;
const int ACS_BUTTON_102 = 102;
const int ACS_BUTTON_103 = 103;
const int ACS_BUTTON_104 = 104;
const int ACS_BUTTON_105 = 105;
const int ACS_BUTTON_106 = 106;
const int ACS_BUTTON_107 = 107;
const int ACS_BUTTON_108 = 108;
const int ACS_BUTTON_109 = 109;
const int ACS_BUTTON_110 = 110;
const int ACS_BUTTON_111 = 111;
const int ACS_BUTTON_112 = 112;
const int ACS_BUTTON_113 = 113;
const int ACS_BUTTON_114 = 114;
const int ACS_BUTTON_115 = 115;
const int ACS_BUTTON_116 = 116;
const int ACS_BUTTON_117 = 117;
const int ACS_BUTTON_118 = 118;
const int ACS_BUTTON_119 = 119;
const int ACS_BUTTON_120 = 120;
const int ACS_BUTTON_121 = 121;
const int ACS_BUTTON_122 = 122;
const int ACS_BUTTON_123 = 123;
const int ACS_BUTTON_124 = 124;
const int ACS_BUTTON_125 = 125;
const int ACS_BUTTON_126 = 126;
const int ACS_BUTTON_127 = 127;
const int ACS_BUTTON_128 = 128;
const int ACS_BUTTON_129 = 129;
const int ACS_BUTTON_130 = 130;
const int ACS_BUTTON_131 = 131;
const int ACS_BUTTON_132 = 132;
const int ACS_BUTTON_133 = 133;
const int ACS_BUTTON_134 = 134;
const int ACS_BUTTON_135 = 135;
const int ACS_BUTTON_136 = 136;
const int ACS_BUTTON_137 = 137;
const int ACS_BUTTON_138 = 138;
const int ACS_BUTTON_139 = 139;
const int ACS_BUTTON_140 = 140;
const int ACS_BUTTON_141 = 141;
const int ACS_BUTTON_142 = 142;
const int ACS_BUTTON_143 = 143;
const int ACS_BUTTON_144 = 144;
const int ACS_BUTTON_145 = 145;
const int ACS_BUTTON_146 = 146;
const int ACS_BUTTON_147 = 147;
const int ACS_BUTTON_148 = 148;
const int ACS_BUTTON_149 = 149;
const int ACS_BUTTON_150 = 150;

const int MAX_ACS_CONTROL_BAR_BUTTONS = 150;

const int SC_POINTER_BUTTON_DOWN = 1;
const int SC_POINTER_MOVE = 2;
const int SC_POINTER_BUTTON_UP = 3;
const int SC_ACS_TOOL_ON = 4;
const int SC_ACS_TOOL_OFF = 5;
const int SC_ACS_BUTTON_ON = 4;
const int SC_ACS_BUTTON_OFF = 5;
```

```
enum ACSReceivePointerEventsEnum
{ ACS_RECEIVE_NO_POINTER_EVENTS = 0
, ACS_RECEIVE_POINTER_EVENTS_WHEN_ACS_BUTTON_ENABLED = 1
, ACS_RECEIVE_POINTER_EVENTS_ALWAYS = 2
, ACS_RECEIVE_POINTER_EVENTS_ALWAYS_FOR_ALL_TOOLS = 3
};
```

```
// These constants for sc.Subgraph[i].AutoColoring
```

```
enum AutoColoringEnum
{ AUTOCOLOR_NONE = 0
, AUTOCOLOR_SLOPE
, AUTOCOLOR_POSNEG
, AUTOCOLOR_BASEGRAPH
, AUTOCOLOR_GRADIENT
, AUTOCOLOR_BASEGRAPH_CLOSE_TO_CLOSE
, NUM_AUTOCOLOR_OPTIONS
};
```

```
enum SubgraphDrawStyles
{ DRAWSTYLE_LINE
, DRAWSTYLE_BAR
, DRAWSTYLE_POINT
, DRAWSTYLE_DASH
, DRAWSTYLE_HIDDEN
, DRAWSTYLE_IGNORE
, DRAWSTYLE_STAIR_STEP
, DRAWSTYLE_SQUARE
, DRAWSTYLE_STAR
, DRAWSTYLE_PLUS
, DRAWSTYLE_ARROW_UP
, DRAWSTYLE_ARROW_DOWN
, DRAWSTYLE_ARROW_LEFT
, DRAWSTYLE_ARROW_RIGHT
, DRAWSTYLE_FILL_TOP
, DRAWSTYLE_FILL_BOTTOM
, DRAWSTYLE_FILL_RECTANGLE_TOP
, DRAWSTYLE_FILL_RECTANGLE_BOTTOM
, DRAWSTYLE_COLOR_BAR
, DRAWSTYLE_BOX_TOP
, DRAWSTYLE_BOX_BOTTOM
, DRAWSTYLE_COLOR_BAR_HOLLOW
, DRAWSTYLE_COLOR_BAR_CANDLE_FILL
, DRAWSTYLE_CUSTOM_TEXT
, DRAWSTYLE_BAR_TOP
, DRAWSTYLE_BAR_BOTTOM
, DRAWSTYLE_LINE_SKIP_ZEROS
, DRAWSTYLE_TRANSPARENT_FILL_TOP
, DRAWSTYLE_TRANSPARENT_FILL_BOTTOM
, DRAWSTYLE_TEXT
, DRAWSTYLE_POINT_ON_LOW
, DRAWSTYLE_POINT_ON_HIGH
, DRAWSTYLE_TRIANGLE_UP
, DRAWSTYLE_TRIANGLE_DOWN
, DRAWSTYLE_TRANSPARENT_FILL_RECTANGLE_TOP
, DRAWSTYLE_TRANSPARENT_FILL_RECTANGLE_BOTTOM
, DRAWSTYLE_BACKGROUND
, DRAWSTYLE_DIAMOND
, DRAWSTYLE_LEFT_PRICE_BAR_DASH
, DRAWSTYLE_RIGHT_PRICE_BAR_DASH
, DRAWSTYLE_TRIANGLE_LEFT
, DRAWSTYLE_TRIANGLE_RIGHT
, DRAWSTYLE_TRIANGLE_RIGHT_OFFSET
, DRAWSTYLE_TRIANGLE_RIGHT_OFFSET_FOR_CANDLESTICK
, DRAWSTYLE_CANDLESTICK_BODY_OPEN
```

```

, DRAWSTYLE_CANDLESTICK_BODY_CLOSE
, DRAWSTYLE_FILL_TO_ZERO
, DRAWSTYLE_TRANSPARENT_FILL_TO_ZERO
, DRAWSTYLE_SQUARE_OFFSET_LEFT
, DRAWSTYLE_SQUARE_OFFSET_LEFT_FOR_CANDLESTICK
, DRAWSTYLE_VALUE_ON_HIGH
, DRAWSTYLE_VALUE_ON_LOW
, DRAWSTYLE_VALUE_OF_SUBGRAPH
, DRAWSTYLE_SUBGRAPH_NAME_AND_VALUE_LABELS_ONLY
, DRAWSTYLE_LINE_AT_LAST_BAR_TO_EDGE
, DRAWSTYLE_FILL_RECTANGLE_TO_ZERO
, DRAWSTYLE_TRANSPARENT_FILL_RECTANGLE_TO_ZERO
, DRAWSTYLE_X
, DRAWSTYLE_CUSTOM_VALUE_AT_Y
, DRAWSTYLE_TRANSPARENT_BAR_TOP
, DRAWSTYLE_TRANSPARENT_BAR_BOTTOM
, DRAWSTYLE_LEFT_OFFSET_BOX_TOP
, DRAWSTYLE_LEFT_OFFSET_BOX_BOTTOM
, DRAWSTYLE_RIGHT_OFFSET_BOX_TOP
, DRAWSTYLE_RIGHT_OFFSET_BOX_BOTTOM
, DRAWSTYLE_HORIZONTAL_PROFILE
, DRAWSTYLE_HORIZONTAL_PROFILE_HOLLOW
, DRAWSTYLE_SQUARE_OFFSET_RIGHT
, DRAWSTYLE_SQUARE_OFFSET_RIGHT_FOR_CANDLESTICK
, DRAWSTYLE_TRANSPARENT_CIRCLE
, DRAWSTYLE_CIRCLE_HOLLOW
, DRAWSTYLE_TRANSPARENT_CIRCLE_VARIABLE_SIZE
, DRAWSTYLE_CIRCLE_HOLLOW_VARIABLE_SIZE
, DRAWSTYLE_POINT_VARIABLE_SIZE
, DRAWSTYLE_LINE_EXTEND_TO_EDGE
, DRAWSTYLE_BACKGROUND_TRANSPARENT
, DRAWSTYLE_CUSTOM_VALUE_AT_Y_WITH_BORDER
, DRAWSTYLE_LEFT_SIDE_TICK_SIZE_RECTANGLE
, DRAWSTYLE_RIGHT_SIDE_TICK_SIZE_RECTANGLE
, DRAWSTYLE_LINE_AT_LAST_BAR_TO_LEFT_SIDE
, DRAWSTYLE_TRANSPARENT_TEXT
, DRAWSTYLE_TEXT_WITH_BACKGROUND
, DRAWSTYLE_TRANSPARENT_CUSTOM_VALUE_AT_Y
, DRAWSTYLE_LEFT_OFFSET_BOX_TOP_FOR_CANDLESTICK
, DRAWSTYLE_LEFT_OFFSET_BOX_BOTTOM_FOR_CANDLESTICK
, DRAWSTYLE_COLOR_BACKGROUND_AT_PRICE
, DRAWSTYLE_RIGHT_OFFSET_BOX_TOP_FOR_CANDLESTICK
, DRAWSTYLE_RIGHT_OFFSET_BOX_BOTTOM_FOR_CANDLESTICK
, DRAWSTYLE_LINE_AT_LAST_BAR_LEFT_TO_RIGHT
, DRAWSTYLE_BOX_TOP_CENTER
, DRAWSTYLE_BOX_BOTTOM_CENTER
, DRAWSTYLE_POINT_VARIABLE_SIZE_WITH_BORDER
, DRAWSTYLE_LINE_FROM_END_OF_CHART_LEFT_TO_RIGHT
, DRAWSTYLE_TRANSPARENT_SQUARE_OFFSET_LEFT_FOR_CANDLESTICK
, DRAWSTYLE_TRANSPARENT_SQUARE_OFFSET_RIGHT_FOR_CANDLESTICK
, DRAWSTYLE_TRANSPARENT_LEFT_SIDE_TICK_SIZE_RECTANGLE
, DRAWSTYLE_TRANSPARENT_RIGHT_SIDE_TICK_SIZE_RECTANGLE
, DRAWSTYLE_TRANSPARENT_CIRCLE_VARIABLE_SIZE_WITH_BORDER
, DRAWSTYLE_LEFT_SIDE_TICK_SIZE_RECTANGLE_OUTLINE
, DRAWSTYLE_RIGHT_SIDE_TICK_SIZE_RECTANGLE_OUTLINE
, DRAWSTYLE_CUSTOM_VALUE_AT_Y_LEFT_ALIGNED
, DRAWSTYLE_CUSTOM_VALUE_AT_Y_RIGHT_ALIGNED
, DRAWSTYLE_STAIR_STEP_TO_EDGE
, DRAWSTYLE_TRANSPARENT_CUSTOM_VALUE_AT_Y_LEFT_ALIGNED
, DRAWSTYLE_TRANSPARENT_CUSTOM_VALUE_AT_Y_RIGHT_ALIGNED
, DRAWSTYLE_POINT_RIGHT_OFFSET
, DRAWSTYLE_POINT_LEFT_OFFSET
, DRAWSTYLE_TRANSPARENT_TEXT_WITH_ALIGNMENT
, NUM_SUBGRAPH_STYLES
};

```



```

#define DRAWSTYLE_STAIR DRAWSTYLE_STAIR_STEP
#define DRAWSTYLE_ARROWUP DRAWSTYLE_ARROW_UP
#define DRAWSTYLE_ARROWDOWN DRAWSTYLE_ARROW_DOWN
#define DRAWSTYLE_ARROWLEFT DRAWSTYLE_ARROW_LEFT
#define DRAWSTYLE_ARROWRIGHT DRAWSTYLE_ARROW_RIGHT
#define DRAWSTYLE_FILLTOP DRAWSTYLE_FILL_TOP
#define DRAWSTYLE_FILLBOTTOM DRAWSTYLE_FILL_BOTTOM
#define DRAWSTYLE_FILLRECTTOP DRAWSTYLE_FILL_RECTANGLE_TOP
#define DRAWSTYLE_FILLRECTBOTTOM DRAWSTYLE_FILL_RECTANGLE_BOTTOM
#define DRAWSTYLE_COLORBAR DRAWSTYLE_COLOR_BAR
#define DRAWSTYLE_BOXTOP DRAWSTYLE_BOX_TOP
#define DRAWSTYLE_BOXBOTTOM DRAWSTYLE_BOX_BOTTOM
#define DRAWSTYLE_COLORBARHOLLOW DRAWSTYLE_COLOR_BAR_HOLLOW
#define DRAWSTYLE_COLORBAR_CANDLEFILL DRAWSTYLE_COLOR_BAR_CANDLE_FILL
#define DRAWSTYLE_BARTOP DRAWSTYLE_BAR_TOP
#define DRAWSTYLE_BARBOTTOM DRAWSTYLE_BAR_BOTTOM
#define DRAWSTYLE_LINE_SKIPZEROS DRAWSTYLE_LINE_SKIP_ZEROS
#define DRAWSTYLE_FILLTOP_TRANSPARENT DRAWSTYLE_TRANSPARENT_FILL_TOP
#define DRAWSTYLE_FILLBOTTOM_TRANSPARENT DRAWSTYLE_TRANSPARENT_FILL_BOTTOM
#define DRAWSTYLE_POINTLOW DRAWSTYLE_POINT_ON_LOW
#define DRAWSTYLE_POINTHIGH DRAWSTYLE_POINT_ON_HIGH
#define DRAWSTYLE_TRIANGLEUP DRAWSTYLE_TRIANGLE_UP
#define DRAWSTYLE_TRIANGLEDOWN DRAWSTYLE_TRIANGLE_DOWN
#define DRAWSTYLE_FILLRECTTOP_TRANSPARENT DRAWSTYLE_TRANSPARENT_FILL_RECTANGLE_TOP
#define DRAWSTYLE_FILLRECTBOTTOM_TRANSPARENT
DRAWSTYLE_TRANSPARENT_FILL_RECTANGLE_BOTTOM
#define DRAWSTYLE_LEFTHASH DRAWSTYLE_LEFT_PRICE_BAR_DASH
#define DRAWSTYLE_RIGHTHASH DRAWSTYLE_RIGHT_PRICE_BAR_DASH
#define DRAWSTYLE_TRIANGLELEFT DRAWSTYLE_TRIANGLE_LEFT
#define DRAWSTYLE_TRIANGLERIGHT DRAWSTYLE_TRIANGLE_RIGHT
#define DRAWSTYLE_TRIANGLERIGHTOFFSET DRAWSTYLE_TRIANGLE_RIGHT_OFFSET
#define DRAWSTYLE_TRIANGLERIGHTOFFSETB DRAWSTYLE_TRIANGLE_RIGHT_OFFSET_FOR_CANDLESTICK
#define DRAWSTYLE_CANDLE_BODYOPEN DRAWSTYLE_CANDLESTICK_BODY_OPEN
#define DRAWSTYLE_CANDLE_BODYCLOSE DRAWSTYLE_CANDLESTICK_BODY_CLOSE
#define DRAWSTYLE_FILLTOZERO DRAWSTYLE_FILL_TO_ZERO
#define DRAWSTYLE_FILLTOZERO_TRANSPARENT DRAWSTYLE_TRANSPARENT_FILL_TO_ZERO
#define DRAWSTYLE_SQUAREOFFSETLEFT DRAWSTYLE_SQUARE_OFFSET_LEFT
#define DRAWSTYLE_SQUARE_OFFSET_LEFT_BODY
DRAWSTYLE_SQUARE_OFFSET_LEFT_FOR_CANDLESTICK
#define DRAWSTYLE_SUBGRAPH_NAME_AND_VALUES_ONLY
DRAWSTYLE_SUBGRAPH_NAME_AND_VALUE_LABELS_ONLY
#define DRAWSTYLE_FILL_RECT_TOZERO DRAWSTYLE_FILL_RECTANGLE_TO_ZERO
#define DRAWSTYLE_FILL_RECT_TOZERO_TRANSPARENT
DRAWSTYLE_TRANSPARENT_FILL_RECTANGLE_TO_ZERO
#define DRAWSTYLE_TRANSPARENT_BARTOP DRAWSTYLE_TRANSPARENT_BAR_TOP
#define DRAWSTYLE_TRANSPARENT_BARBOTTOM DRAWSTYLE_TRANSPARENT_BAR_BOTTOM
#define DRAWSTYLE_LEFT_OFFSET_BOXTOP DRAWSTYLE_LEFT_OFFSET_BOX_TOP
#define DRAWSTYLE_LEFT_OFFSET_BOXBOTTOM DRAWSTYLE_LEFT_OFFSET_BOX_BOTTOM
#define DRAWSTYLE_RIGHT_OFFSET_BOXTOP DRAWSTYLE_RIGHT_OFFSET_BOX_TOP
#define DRAWSTYLE_RIGHT_OFFSET_BOXBOTTOM DRAWSTYLE_RIGHT_OFFSET_BOX_BOTTOM
#define DRAWSTYLE_SQUARE_OFFSET_RIGHT_BODY
DRAWSTYLE_SQUARE_OFFSET_RIGHT_FOR_CANDLESTICK

```

// Note: This is written to disk, so the size and values must not change.

```

enum SubgraphLineStyle : int16_t
{
    LINESTYLE_UNSET = -1
    , LINESTYLE_SOLID = 0
    , LINESTYLE_DASH = 1
    , LINESTYLE_DOT = 2
    , LINESTYLE_DASHDOT = 3
    , LINESTYLE_DASHDOTDOT = 4
    , LINESTYLE_ALTERNATE = 5
};

```



```

/*=====*/
inline bool IsValidSubgraphLineStyle(const int16_t SubgraphLineStyle)
{
    switch (SubgraphLineStyle)
    {
        case LINESSTYLE_SOLID:
        case LINESSTYLE_DASH:
        case LINESSTYLE_DOT:
        case LINESSTYLE_DASHDOT:
        case LINESSTYLE_DASHDOTDOT:
        case LINESSTYLE_ALTERNATE:
            return true;

        default:
            return false;
    }
}

/*=====*/
inline SubgraphLineStyles SubgraphLineStyleFromInt
( const int SubgraphLineStyle
, const SubgraphLineStyles Default = LINESSTYLE_SOLID
)
{
    switch (SubgraphLineStyle)
    {
        case LINESSTYLE_SOLID:
            return LINESSTYLE_SOLID;

        case LINESSTYLE_DASH:
            return LINESSTYLE_DASH;

        case LINESSTYLE_DOT:
            return LINESSTYLE_DOT;

        case LINESSTYLE_DASHDOT:
            return LINESSTYLE_DASHDOT;

        case LINESSTYLE_DASHDOTDOT:
            return LINESSTYLE_DASHDOTDOT;

        case LINESSTYLE_ALTERNATE:
            return LINESSTYLE_ALTERNATE;

        default:
            return Default;
    }
}

enum SubgraphLineLabelFlags
{ LL_DISPLAY_NAME           = 0x00000001
, LL_NAME_ALIGN_CENTER     = 0x00000002
, LL_NAME_ALIGN_FAR_RIGHT  = 0x00000004
, LL_NAME_ALIGN_ABOVE      = 0x00000008
, LL_NAME_ALIGN_BELOW      = 0x00000010
, LL_NAME_ALIGN_RIGHT      = 0x00000020
, LL_NAME_ALIGN_VALUES_SCALE = 0x00000040
, LL_NAME_ALIGN_LEFT_EDGE  = 0x00000080
, LL_DISPLAY_VALUE         = 0x00000100
, LL_VALUE_ALIGN_CENTER    = 0x00000200
, LL_VALUE_ALIGN_FAR_RIGHT = 0x00000400
, LL_VALUE_ALIGN_ABOVE     = 0x00000800
, LL_VALUE_ALIGN_BELOW     = 0x00001000
, LL_VALUE_ALIGN_RIGHT     = 0x00002000

```

```
, LL_VALUE_ALIGN_VALUES_SCALE = 0x00004000
, LL_VALUE_ALIGN_LEFT_EDGE    = 0x00008000
, LL_NAME_ALIGN_LEFT          = 0x00010000
, LL_VALUE_ALIGN_LEFT         = 0x00020000
, LL_NAME_REVERSE_COLORS      = 0x00040000
, LL_VALUE_REVERSE_COLORS_INV = 0x00080000 // This flag is inverted so that the default value of 0 means it is
enabled.
, LL_NAME_ALIGN_DOM_LABELS_COLUMN = 0x00100000
, LL_VALUE_ALIGN_DOM_LABELS_COLUMN = 0x00200000
, LL_DISPLAY_CUSTOM_VALUE_AT_Y = 0x00400000//LL_VALUE_* flags also applies to this value label
, LL_NAME_ALIGN_LEFT_SIDE_VALUES_SCALE = 0x00800000
, LL_VALUE_ALIGN_LEFT_SIDE_VALUES_SCALE = 0x01000000
};
```

```
enum SubgraphNameAndValueDisplayFlags
{ SNV_DISPLAY_IN_WINDOWS = 0x00000001
, SNV_DISPLAY_IN_DATA_LINE = 0x00000002
};
```

```
enum ChartDataTypeEnum : int32_t
{ NO_DATA_TYPE = 0
, DAILY_DATA = 1
, INTRADAY_DATA = 2
, MARKET_DEPTH_DATA = 3
};
```

```
enum ScaleTypeEnum
{ SCALE_LINEAR = 0
, SCALE_LOGARITHMIC = 1
};
```

```
enum ScaleRangeTypeEnum : unsigned int
{ SCALE_AUTO = 0
, SCALE_USERDEFINED = 1
, SCALE_INDEPENDENT = 2
, SCALE_SAMEASREGION = 3
, SCALE_CONSTANT_RANGE = 4
, SCALE_CONSTRANGECENTER = 5
// Do not use SCALE_CONSTRANGECENTER -- instead use SCALE_CONSTANTRANGE
// with CONST_RANGE_MODE_AUTO_CENTER_LAST_BAR.
, SCALE_ZEROBASED = 6
, SCALE_ZEROCENTERED = 7
};
```

```
/*=====
Sets the given r_ScaleRangeType according to the given Value, so long as
the given Value is valid. If the given Value is not valid, the given
r_ScaleRangeType is left unchanged.
-----*/
```

```
inline void SetScaleRangeTypeFromInt
( ScaleRangeTypeEnum& r_ScaleRangeType
, int Value
)
{
    switch (Value)
    {
        case SCALE_AUTO:
            r_ScaleRangeType = SCALE_AUTO;
            break;

        case SCALE_USERDEFINED:
            r_ScaleRangeType = SCALE_USERDEFINED;
            break;
```

```

    case SCALE_INDEPENDENT:
        r_ScaleRangeType = SCALE_INDEPENDENT;
        break;

    case SCALE_SAMEASREGION:
        r_ScaleRangeType = SCALE_SAMEASREGION;
        break;

    case SCALE_CONSTANT_RANGE:
        r_ScaleRangeType = SCALE_CONSTANT_RANGE;
        break;

    case SCALE_CONSTRANGECENTER:
        r_ScaleRangeType = SCALE_CONSTRANGECENTER;
        break;

    case SCALE_ZEROBASED:
        r_ScaleRangeType = SCALE_ZEROBASED;
        break;

    case SCALE_ZEROCENTERED:
        r_ScaleRangeType = SCALE_ZEROCENTERED;
        break;
}
}

enum ScaleConstantRangeModeEnum : int
{
    CONST_RANGE_MODE_MANUAL = 0
    , CONST_RANGE_MODE_AUTO_CENTER_LAST_BAR = 1
    , CONST_RANGE_MODE_AUTO_CENTER_LAST_PRICE = 2
    , CONST_RANGE_MODE_AUTO_KEEP_LAST_BAR_WITHIN_TICKS_FROM_EDGE = 3
    , CONST_RANGE_MODE_AUTO_CENTER_WHEN_BAR_BEYOND_TICKS_FROM_CENTER = 4
    , CONST_RANGE_MODE_AUTO_CENTER_WHEN_PRICE_BEYOND_TICKS_FROM_CENTER = 5
};

/*=====
Returns the name of the given ScaleConstantRangeMode. Returns an empty
string if the given mode is not valid.
-----*/
inline const char* GetScaleConstantRangeModeName
( const ScaleConstantRangeModeEnum ConstantRangeMode
)
{
    switch (ConstantRangeMode)
    {
        case CONST_RANGE_MODE_MANUAL:
            return "Manual Mode";

        case CONST_RANGE_MODE_AUTO_CENTER_LAST_BAR:
            return "Re-Center To The Last Bar On Every Update";

        case CONST_RANGE_MODE_AUTO_CENTER_LAST_PRICE:
            return "Re-Center To The Last Price On Every Update";

        case CONST_RANGE_MODE_AUTO_KEEP_LAST_BAR_WITHIN_TICKS_FROM_EDGE:
            return "Keep The Last Bar Within View";

        case CONST_RANGE_MODE_AUTO_CENTER_WHEN_BAR_BEYOND_TICKS_FROM_CENTER:
            return "Re-Center When The Last Bar Exceeds A Certain Number Of Ticks From The Center";

        case CONST_RANGE_MODE_AUTO_CENTER_WHEN_PRICE_BEYOND_TICKS_FROM_CENTER:
            return "Re-Center When The Last Price Exceeds A Certain Number Of Ticks From The Center";

        default:
            return "";
    }
}

```

```

    }
}

/*=====
Sets the given r_ConstantRangeMode according to the given Value, so long
as the given Value is valid. If the given Value is not valid, the given
r_ConstantRangeMode is left unchanged.
-----*/
inline void SetScaleConstantRangeModeFromInt
( ScaleConstantRangeModeEnum& r_ConstantRangeMode
, int Value
)
{
    switch (Value)
    {
        case CONST_RANGE_MODE_MANUAL:
            r_ConstantRangeMode = CONST_RANGE_MODE_MANUAL;
            break;

        case CONST_RANGE_MODE_AUTO_CENTER_LAST_BAR:
            r_ConstantRangeMode = CONST_RANGE_MODE_AUTO_CENTER_LAST_BAR;
            break;

        case CONST_RANGE_MODE_AUTO_CENTER_LAST_PRICE:
            r_ConstantRangeMode = CONST_RANGE_MODE_AUTO_CENTER_LAST_PRICE;
            break;

        case CONST_RANGE_MODE_AUTO_KEEP_LAST_BAR_WITHIN_TICKS_FROM_EDGE:
        {
            r_ConstantRangeMode
                = CONST_RANGE_MODE_AUTO_KEEP_LAST_BAR_WITHIN_TICKS_FROM_EDGE;
        }
            break;

        case CONST_RANGE_MODE_AUTO_CENTER_WHEN_BAR_BEYOND_TICKS_FROM_CENTER:
        {
            r_ConstantRangeMode
                = CONST_RANGE_MODE_AUTO_CENTER_WHEN_BAR_BEYOND_TICKS_FROM_CENTER;
        }
            break;

        case CONST_RANGE_MODE_AUTO_CENTER_WHEN_PRICE_BEYOND_TICKS_FROM_CENTER:
        {
            r_ConstantRangeMode
                = CONST_RANGE_MODE_AUTO_CENTER_WHEN_PRICE_BEYOND_TICKS_FROM_CENTER;
        }
            break;
    }
}

enum MovAvgTypeEnum
{ MOVAVGTYPE_EXPONENTIAL
, MOVAVGTYPE_LINEARREGRESSION
, MOVAVGTYPE_SIMPLE
, MOVAVGTYPE_WEIGHTED
, MOVAVGTYPE_WILDERS
, MOVAVGTYPE_SIMPLE_SKIP_ZEROS
, MOVAVGTYPE_SMOOTHED
, MOVAVGTYPE_NUMBER_OF_AVERAGES
};

enum
{ BHCS_BAR_HAS_CLOSED = 2
, BHCS_BAR_HAS_NOT_CLOSED = 3
, BHCS_SET_DEFAULTS = 4

```

```

};

// These constants are used to set the calculation precedence through the
// sc.CalculationPrecedence member variable.
enum PrecedenceLevelEnum
{ STD_PREC_LEVEL = 0
, LOW_PREC_LEVEL = 1
, VERY_LOW_PREC_LEVEL = 2
};

enum UseToolAddMethodEnum
{ UTAM_NOT_SET
, UTAM_ADD_ALWAYS
, UTAM_ADD_ONLY_IF_NEW
, UTAM_ADD_OR_ADJUST
};

// Constants for s_TimeAndSales record Level:
const int SC_TS_MARKER = 0; // Indicates a gap in the time and sales data
const int SC_TS_BID = 1; // Trade is considered to have occurred at Bid price or lower
const int SC_TS_ASK = 2; // Trade is considered to have occurred at Ask price or higher
const int SC_TS_BIDASKVALUES = 6; // Bid and Ask quote data update

const int COUNTDOWN_TIMER_CHART_DRAWINGNUMBER = 40338335;
const int SYMBOL_TEXT_DRAWING_NUMBER = 40338336;

enum InputValueTypes
{ NO_VALUE
, OHLC_VALUE // IndexValue
, FLOAT_VALUE // FloatValue
, STUDYINDEX_VALUE // IndexValue
, SUBGRAPHINDEX_VALUE // IndexValue
, YESNO_VALUE // BooleanValue
, MOVAVGTYPE_VALUE // IndexValue
, UNUSED7
, DATE_VALUE // DateTimeValue
, TIME_VALUE // DateTimeValue
, DATETIME_VALUE // DateTimeValue
, INT_VALUE // IntValue
, UNUSED12
, STUDYID_VALUE // IndexValue
, COLOR_VALUE // ColorValue
, ALERT_SOUND_NUMBER_VALUE // IndexValue (0 = No Alert, 1 = Alert 1, 2 = Alert 2, ...)
, CANDLESTICK_PATTERNS_VALUE // IndexValue
, TIME_PERIOD_LENGTH_UNIT_VALUE // IndexValue
, CHART_STUDY_SUBGRAPH_VALUES // ChartNum, StudyID, SubgraphIndex
, CHART_NUMBER // IntValue
, STUDY_SUBGRAPH_VALUES // StudyID, SubgraphIndex
, CHART_STUDY_VALUES // ChartNum, StudyID
, CUSTOM_STRING_VALUE // IndexValue
, DOUBLE_VALUE // DoubleValue
, TIMEZONE_VALUE // IndexValue
, TIME_WITH_TIMEZONE_VALUE // DateTimeValue, TimeZoneValue
, STRING_VALUE // StringValue
, PATH_AND_FILE_NAME_VALUE // StringValue
, FIND_SYMBOL_VALUE // StringValue
};

enum EnumTimePeriodLengthUnit
{ TIME_PERIOD_LENGTH_UNIT_MINUTES
, TIME_PERIOD_LENGTH_UNIT_DAYS
, TIME_PERIOD_LENGTH_UNIT_WEEKS

```

```

, TIME_PERIOD_LENGTH_UNIT_MONTHS
, TIME_PERIOD_LENGTH_UNIT_YEARS
, TIME_PERIOD_LENGTH_UNIT_SECONDS
, TIME_PERIOD_LENGTH_UNIT_QUARTERS
, TIME_PERIOD_LENGTH_UNIT_COUNT
};

/*=====*/
inline const char* TimePeriodLengthUnitToString(int TimePeriodLengthUnit)
{
    switch (TimePeriodLengthUnit)
    {
        case TIME_PERIOD_LENGTH_UNIT_SECONDS:
            return "Seconds";

        case TIME_PERIOD_LENGTH_UNIT_MINUTES:
            return "Minutes";

        case TIME_PERIOD_LENGTH_UNIT_DAYS:
            return "Days";

        case TIME_PERIOD_LENGTH_UNIT_WEEKS:
            return "Weeks";

        case TIME_PERIOD_LENGTH_UNIT_MONTHS:
            return "Months";

        case TIME_PERIOD_LENGTH_UNIT_QUARTERS:
            return "Quarters";

        case TIME_PERIOD_LENGTH_UNIT_YEARS:
            return "Years";

        default:
            return "";
    }
}

// These constants are for the CrossOver function
const int CROSS_FROM_TOP = 1;
const int CROSS_FROM_BOTTOM = -1;
const int NO_CROSS = 0;

enum OperatorEnum
{ NOT_EQUAL_OPERATOR
, LESS_EQUAL_OPERATOR
, GREATER_EQUAL_OPERATOR
, CROSS_EQUAL_OPERATOR
, CROSS_OPERATOR
, EQUAL_OPERATOR
, LESS_OPERATOR
, GREATER_OPERATOR
, CROSS_FROM_BELOW_OPERATOR
, CROSS_FROM_ABOVE_OPERATOR

, NUM_OPERATORS
};

const int SC_SUBGRAPHS_AVAILABLE = 60;
const int MAX_STUDY_SUBGRAPHS = 60;
const int MAX_SUBGRAPH_EXTRA_ARRAYS = 12;
const int SC_INPUTS_AVAILABLE = 128;

```

```

const int SC_DRAWING_MAX_LEVELS_OLD = 32;
const int SC_DRAWING_MAX_LEVELS = 64; // Do not change this, used in s_ChartDrawing and s_ConfigData
const int ACSIL_DRAWING_MAX_LEVELS = 32;

const int NUM_LINREG_TOOL_STD_DEVIATIONS = 12;

// Constance for GraphDrawType.
// These values are written to a file and must not change for individual Graph Draw Type.
enum GraphDrawTypeEnum
{
    GDT_CUSTOM = 0
    , GDT_OHLCBAR = 1
    , GDT_CANDLESTICK = 2
    , GDT_CANDLESTICK_BODY_ONLY = 3
    , GDT_LINEONCLOSE = 4
    , GDT_MOUNTAIN = 5
    , GDT_HLCBAR = 6
    , GDT_LINEONOPEN = 7
    , GDT_LINEONHLAVG = 8
    , GDT_STAIRSTEPONCLOSE = 9
    , GDT_HLBAR = 10
    , GDT_TPO_PROFILE = 11
    , GDT_KAGI = 12
    , GDT_BOX_UNUSED = 13
    , GDT_POINT_AND_FIGURE_BARS = 14
    , GDT_VOLUME_BY_PRICE = 15
    , GDT_POINT_AND_FIGURE_XO = 16
    , GDT_BID_ASK_BAR = 17
    , GDT_PRICE_VOLUME = 18
    , GDT_CANDLE_PRICE_VOLUME_BAR = 19
    // Values 20 - 24 were used previously.
    , GDT_BLANK = 25
    , GDT_NUMBERS_BARS = 26
    , GDT_NUMERIC_INFORMATION = 27
    , GDT_RENKO_BRICK = 28
    , GDT_RENKO_BRICK_WITH_WICKS = 29
    , GDT_CANDLESTICK_HOLLOW = 30
    , GDT_MARKET_DEPTH = 31
    , GDT_VOLUME_LEVEL_TRADES = 32
    , GDT_CANDLE_PRICE_VOLUME_BAR_HOLLOW = 33

    // if new types are added, be sure to update GRAPHTYPE_STRING
    , NUM_GRAPH_DRAW_TYPES
};

enum ValueFormatEnum : int
{
    VALUEFORMAT_UNSET = -1

    , VALUEFORMAT_WHOLE_NUMBER = 0

    , VALUEFORMAT_1_DIGIT = 1
    , VALUEFORMAT_2_DIGITS = 2
    , VALUEFORMAT_3_DIGITS = 3
    , VALUEFORMAT_4_DIGITS = 4
    , VALUEFORMAT_5_DIGITS = 5
    , VALUEFORMAT_6_DIGITS = 6
    , VALUEFORMAT_7_DIGITS = 7
    , VALUEFORMAT_8_DIGITS = 8
    , VALUEFORMAT_9_DIGITS = 9

    , VALUEFORMAT_MAX_DIGITS = 19

    , VALUEFORMAT_TIME = 20
    , VALUEFORMAT_DATE = 21
    , VALUEFORMAT_DATETIME = 22

```



```
, VALUEFORMAT_CURRENCY = 23
, VALUEFORMAT_PERCENT = 24
, VALUEFORMAT_SCIENTIFIC = 25
, VALUEFORMAT_LARGE_INTEGER_WITH_SUFFIX = 26
, VALUEFORMAT_BOOLEAN = 27
```

```
, VALUEFORMAT_INHERITED = 40
```

```
, VALUEFORMAT_1_2 = 102
, VALUEFORMAT_1_4 = 104
, VALUEFORMAT_1_8 = 108
, VALUEFORMAT_1_16 = 116
, VALUEFORMAT_1_32 = 132
, VALUEFORMAT_D5_32 = 134
, VALUEFORMAT_D25_32 = 136
, VALUEFORMAT_D125_32 = 140
, VALUEFORMAT_1_64 = 164
, VALUEFORMAT_1_128 = 228
, VALUEFORMAT_1_256 = 356
};
```

```
const int VALUEFORMAT_2_DENOMINATOR = VALUEFORMAT_1_2;
```

```
//Server connection state constants. These relate to File >> Connect to Data Feed.
```

```
enum ServerConnectionStateEnum
{ SCS_DISCONNECTED = 0
, SCS_CONNECTING = 1
, SCS_CONNECTED = 2
, SCS_CONNECTION_LOST = 3
, SCS_DISCONNECTING = 4
, SCS_RECONNECTING = 5
};
```

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

```
enum ProfitLossDisplayFormatEnum
{ PPLD_DO_NOT_DISPLAY = 0
, PPLD_CURRENCY_VALUE
, PPLD_POINTS
, PPLD_POINTS_IGNORE_QUANTITY
, PPLD_TICKS
, PPLD_TICKS_IGNORE_QUANTITY
, PPLD_CV_AND_POINTS
, PPLD_CV_AND_POINTS_IGNORE_QUANTITY
, PPLD_CV_AND_TICKS
, PPLD_CV_AND_TICKS_IGNORE_QUANTITY
};
```

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

```
// Time Zones
```

```
enum TimeZonesEnum
{ TIMEZONE_NOT_SPECIFIED = 0
, TIMEZONE_HONOLULU // Hawaii
, TIMEZONE_ANCHORAGE // Alaska
, TIMEZONE_LOS_ANGELES // US Pacific
, TIMEZONE_PHOENIX_ARIZONA
, TIMEZONE_DENVER // US Mountain
, TIMEZONE_CHICAGO // US Central
, TIMEZONE_NEW_YORK // US Eastern
, TIMEZONE_HALIFAX
, TIMEZONE_UTC
, TIMEZONE_LONDON
, TIMEZONE_BRUSSELS
, TIMEZONE_CAPE_TOWN
, TIMEZONE_ATHENS
```

```
, TIMEZONE_BAGHDAD
, TIMEZONE_MOSCOW
, TIMEZONE_MOSCOW_2011_TO_2014
, TIMEZONE_DUBAI
, TIMEZONE_ISLAMABAD
, TIMEZONE_NEW_DELHI
, TIMEZONE_DHAKA
, TIMEZONE_JAKARTA
, TIMEZONE_HONG_KONG
, TIMEZONE_TOKYO
, TIMEZONE_BRISBANE
, TIMEZONE_SYDNEY
, TIMEZONE.UTC_PLUS_12
, TIMEZONE_AUCKLAND
, TIMEZONE_CUSTOM

, NUM_TIME_ZONES
};
```

```
static const char* TIME_ZONE_POSIX_STRINGS[NUM_TIME_ZONES] =
{ "" // Not specified
, "HST-10" // Honolulu
, "AKST-09AKDT+01,M3.2.0/02:00,M11.1.0/02:00" // Anchorage - DST rule for 2007+
, "PST-08PDT+01,M3.2.0/02:00,M11.1.0/02:00" // Los Angeles - DST rule for 2007+
, "MST-07" // Phoenix Arizona
, "MST-07MDT+01,M3.2.0/02:00,M11.1.0/02:00" // Denver - DST rule for 2007+
, "CST-06CDT+01,M3.2.0/02:00,M11.1.0/02:00" // Chicago - DST rule for 2007+
, "EST-05EDT+01,M3.2.0/02:00,M11.1.0/02:00" // New York - DST rule for 2007+
, "AST-04ADT+01,M3.2.0/02:00,M11.1.0/02:00" // Halifax - DST rule for 2007+
, "UTC+00" // UTC (Reykjavik)
, "GMT+00BST+01,M3.5.0/01:00,M10.5.0/02:00" // London
, "CET+01CEST+01,M3.5.0/02:00,M10.5.0/03:00" // Brussels
, "SAST+02" // Cape Town
, "EET+02EEST+01,M3.5.0/03:00,M10.5.0/04:00" // Athens
, "AST+03" // Baghdad
, "MSK+03" // Moscow
, "MSK+04" // Moscow 2011-2014 (before summer 2011: MSK+03MSD+01,M3.5.0/02:00,M10.5.0/02:00)
, "GST+04" // Dubai
, "PKT+05" // Islamabad
, "IST+05:30" // New Delhi
, "BDT+06" // Dhaka
, "WIT+07" // Jakarta
, "HKT+08" // Hong Kong
, "JST+09" // Tokyo
, "AEST+10" // Brisbane
, "AEST+10AEDT+01,M10.1.0/02:00,M4.1.0/03:00" // Sydney
, "UTC+12"
, "NZST+12NZDT+01,M9.5.0/02:00,M4.1.0/03:00" // Auckland - DST rule for Sep. 2007+
, "" // Other
};
```

```
static const char* TIME_ZONE_NAME_STRINGS[NUM_TIME_ZONES] =
{ "Not Specified"
, "Honolulu (-10 HST)"
, "Anchorage (-9 AKST/-8 AKDT)"
, "Los Angeles (-8 PST/-7 PDT)"
, "Phoenix (-7 MST)"
, "Denver (-7 MST/-6 MDT)"
, "Chicago (-6 CST/-5 CDT)"
, "New York (-5 EST/-4 EDT)"
, "Halifax (-4 AST/-3 ADT)"
, "UTC (+0 UTC)"
, "London (+0 GMT/+1 BST)"
, "Brussels (+1 CET/+2 CEST)"
};
```

```

, "Cape Town (+2 SAST)"
, "Athens (+2 EET/+3 EEST)"
, "Baghdad (+3 AST)"
, "Moscow (+3 MSK)"
, "Moscow 2011-2014 (+4 MSK)"
, "Dubai (+4 GST)"
, "Islamabad (+5 PKT)"
, "New Delhi (+5:30 IST)"
, "Dhaka (+6 BDT)"
, "Jakarta (+7 WIT)"
, "Hong Kong (+8 HKT)"
, "Tokyo (+9 JST)"
, "Brisbane (+10 AEST)"
, "Sydney (+10 AEST/+11 AEDT)"
, "UTC +12 (+12 UTC)"
, "Auckland (+12 NZST/+13 NZDT)"
, "Other/Custom DST..."
};

static const char* TIME_ZONE_SHORT_NAME_STRINGS[NUM_TIME_ZONES] =
{
    "N/A"
, "HST"
, "AKST/AKDT"
, "PST/PDT"
, "MST"
, "MST/MDT"
, "CST/CDT"
, "EST/EDT"
, "AST/ADT"
, "UTC"
, "GMT/BST"
, "CET/CEST"
, "EET/EEST"
, "AST"
, "MSK"
, "MSK 2011"
, "GST"
, "PKT"
, "IST"
, "BDT"
, "WIT"
, "HKT"
, "JST"
, "AEST"
, "AEST/AEDT"
, "UTC+12"
, "NZST/NZDT"
, "Custom"
};

/*=====*/
inline const char* GetPosixStringForTimeZone(TimeZonesEnum TimeZone)
{
    if (TimeZone < TIMEZONE_NOT_SPECIFIED || TimeZone >= NUM_TIME_ZONES)
        return TIME_ZONE_POSIX_STRINGS[TIMEZONE_NOT_SPECIFIED];

    return TIME_ZONE_POSIX_STRINGS[TimeZone];
}

/*=====*/
inline TimeZonesEnum GetTimeZoneFromPosixString(const char* PosixString)
{
    if (PosixString == NULL)
        return TIMEZONE_NOT_SPECIFIED;

    for (int TimeZoneIndex = TIMEZONE_NOT_SPECIFIED; TimeZoneIndex < NUM_TIME_ZONES; ++TimeZoneIndex)

```

```

{
    #if _MSC_VER >= 1400
    if (_stricmp(PosixString, TIME_ZONE_POSIX_STRINGS[TimeZoneIndex]) == 0)
    #else
    if (stricmp(PosixString, TIME_ZONE_POSIX_STRINGS[TimeZoneIndex]) == 0)
    #endif
        return static_cast<TimeZonesEnum>(TimeZoneIndex);
}

return TIMEZONE_NOT_SPECIFIED;
}

/*****

enum IntradayBarPeriodTypeEnum : int32_t
{
    IBPT_DAYS_MINS_SECS = 0
    , IBPT_VOLUME_PER_BAR = 1
    , IBPT_NUM_TRADES_PER_BAR = 2
    , IBPT_RANGE_IN_TICKS_STANDARD = 3
    , IBPT_RANGE_IN_TICKS_NEWBAR_ON_RANGEMET = 4
    , IBPT_RANGE_IN_TICKS_TRUE = 5
    , IBPT_RANGE_IN_TICKS_FILL_GAPS = 6
    , IBPT_REVERSAL_IN_TICKS = 7
    , IBPT_RENKO_IN_TICKS = 8
    , IBPT_DELTA_VOLUME_PER_BAR = 9
    , IBPT_FLEX_RENKO_IN_TICKS = 10
    , IBPT_RANGE_IN_TICKS_OPEN_EQUAL_CLOSE = 11
    , IBPT_PRICE_CHANGES_PER_BAR = 12
    , IBPT_MONTHS_PER_BAR = 13
    , IBPT_POINT_AND_FIGURE = 14
    , IBPT_FLEX_RENKO_IN_TICKS_INVERSE_SETTINGS = 15
    , IBPT_ALIGNED_RENKO = 16
    , IBPT_RANGE_IN_TICKS_NEW_BAR_ON_RANGE_MET_OPEN_EQUALS_PRIOR_CLOSE = 17
    , IBPT_ACSIL_CUSTOM = 18
    , IBPT_HISTORICAL_DAILY_DATA_DAYS_PER_BAR = 10000
};

// Useful constants for HTTP requests
const int HTTP_REQUEST_ERROR = 0;
const int HTTP_REQUEST_NOT_SENT = 1;
const int HTTP_REQUEST_MADE = 2;
const int HTTP_REQUEST_RECEIVED = 3;

const char* const ACSIL_HTTP_REQUEST_ERROR_TEXT = "HTTP_REQUEST_ERROR";
const char* const ACSIL_HTTP_EMPTY_RESPONSE_TEXT = "EMPTY_RESPONSE";

// Trading related error constants

const int SCTRADING_ORDER_ERROR = -1;
const int SCTRADING_NOT_OCO_ORDER_TYPE = -2;
const int SCTRADING_ATTACHED_ORDER_OFFSET_NOT_SUPPORTED_WITH_MARKET_PARENT = -3;
const int SCTRADING_UNSUPPORTED_ATTACHED_ORDER = -4;
const int SCTRADING_SYMBOL_SETTINGS_NOT_FOUND = -5;
const int ACSIL_GENERAL_NULL_POINTER_ERROR = -6;
//-7
const int SCTRADING_UNSUPPORTED_ORDER_TYPE = -8;
const int SCTRADING_ERROR_SETTING_ORDER_PRICES = -9;

const int SCT_SKIPPED_DOWNLOADING_HISTORICAL_DATA = -8999;
const int SCT_SKIPPED_FULL_RECALC = -8998;
const int SCT_SKIPPED_ONLY_ONE_TRADE_PER_BAR = -8997;
const int SCT_SKIPPED_INVALID_INDEX_SPECIFIED = -8996;
const int SCT_SKIPPED_TOO_MANY_NEW_BARS_DURING_UPDATE = -8995;
const int SCT_SKIPPED_AUTO_TRADING_DISABLED = -8994;

```

```

const int SCT_ORDERTYPE_MARKET = 0;
const int SCT_ORDERTYPE_LIMIT = 1;
const int SCT_ORDERTYPE_STOP = 2;
const int SCT_ORDERTYPE_STOP_LIMIT = 3;
const int SCT_ORDERTYPE_MARKET_IF_TOUCHED = 4;
const int SCT_ORDERTYPE_LIMIT_CHASE = 5;
const int SCT_ORDERTYPE_LIMIT_TOUCH_CHASE = 6;
const int SCT_ORDERTYPE_TRAILING_STOP = 7;
const int SCT_ORDERTYPE_TRAILING_STOP_LIMIT = 8;
const int SCT_ORDERTYPE_TRIGGERED_TRAILING_STOP_3_OFFSETS = 9;
const int SCT_ORDERTYPE_TRIGGERED_TRAILING_STOP_LIMIT_3_OFFSETS = 10;
const int SCT_ORDERTYPE_STEP_TRAILING_STOP = 11;
const int SCT_ORDERTYPE_STEP_TRAILING_STOP_LIMIT = 12;
const int SCT_ORDERTYPE_TRIGGERED_STEP_TRAILING_STOP = 13;
const int SCT_ORDERTYPE_TRIGGERED_STEP_TRAILING_STOP_LIMIT = 14;
const int SCT_ORDERTYPE_OCO_LIMIT_STOP = 15;
const int SCT_ORDERTYPE_OCO_LIMIT_STOP_LIMIT = 16;
const int SCT_ORDERTYPE_OCO_BUY_STOP_SELL_STOP = 17;
const int SCT_ORDERTYPE_OCO_BUY_STOP_LIMIT_SELL_STOP_LIMIT = 18;
const int SCT_ORDERTYPE_OCO_BUY_LIMIT_SELL_LIMIT = 19;
const int SCT_ORDERTYPE_LIMIT_IF_TOUCHED = 20;
const int SCT_ORDERTYPE_BID_ASK_QUANTITY_TRIGGERED_STOP = 21;
const int SCT_ORDERTYPE_TRIGGERED_LIMIT = 22;
const int SCT_ORDERTYPE_TRADE_VOLUME_TRIGGERED_STOP = 23;
const int SCT_ORDERTYPE_STOP_WITH_BID_ASK_TRIGGERING = 24;
const int SCT_ORDERTYPE_STOP_WITH_LAST_TRIGGERING = 25;
const int SCT_ORDERTYPE_LIMIT_IF_TOUCHED_CLIENT_SIDE = 26;
const int SCT_ORDERTYPE_MARKET_IF_TOUCHED_CLIENT_SIDE = 27;
const int SCT_ORDERTYPE_TRADE_VOLUME_TRIGGERED_STOP_LIMIT = 28;
const int SCT_ORDERTYPE_STOP_LIMIT_CLIENT_SIDE = 29;
const int SCT_ORDERTYPE_TRIGGERED_STOP = 30;
const int SCT_ORDERTYPE_TRIGGERED_LIMIT_WITH_CHASE = 31;
const int SCT_ORDERTYPE_MARKET_LIMIT = 32;
const int SCT_ORDERTYPE_NUM_ORDERS = 33;

```

```

enum SCTimeInForceEnum
{
    SCT_TIF_UNSET = -1
    , SCT_TIF_DAY = 0
    , SCT_TIF_GTC = 1//good till canceled
    , SCT_TIF_GOOD_TILL_CANCELED = 1//good till canceled
    , SCT_TIF_IMMEDIATE_OR_CANCEL = 2
    , SCT_TIF_FILL_OR_KILL = 3
    , SCT_TIF_ALL_OR_NONE = 4
};

```

// Trading order status codes. SCT\_ = Sierra Chart Trading. OSC\_ = OrderStatusCode.  
 // These codes are saved to a file and existing codes must not change in value.

```

enum SCOrderStatusCodeEnum
{ SCT_OSC_UNSPECIFIED = 0 //OrderStatusCode is undefined. You will never get this. It is internally used.

    , SCT_OSC_ORDERSENT = 1
    , SCT_OSC_PENDINGOPEN = 2 // This is only used for external trading services, and it should be used when the service
    indicates the order is pending and it is clear that the order cannot be modified or it is unclear if the order can be modified
    in the pending state. Sierra Chart will queue and delay an order modification if the state is Pending Open. This order
    status must not be used in the case of child orders or when an order like a Market If Touched order is being held until it is
    triggered.

    , SCT_OSC_PENDING_CHILD_CLIENT = 3 //Indicates a pending child order that resides on the client-side within Sierra
    Chart.
    , SCT_OSC_PENDING_CHILD_SERVER = 4 //Indicates a pending child order that resides on the server.
    , SCT_OSC_OPEN = 5
    , SCT_OSC_PENDINGMODIFY = 6

```

```
, SCT_OSC_PENDINGCANCEL = 7
, SCT_OSC_FILLED = 8 // Order has been completely filled or partially filled, and is no longer considered working.
, SCT_OSC_CANCELED = 9 // Order has been canceled. The order could have been partially filled before it was
canceled.
, SCT_OSC_ERROR = 10 // Error with order.
, SCT_OSC_PENDING_CANCEL_FOR_REPLACE = 11 //To be used with trading services that do not support modifying
orders.
, SCT_OSC_HELD = 12 //This is used internally by Sierra Chart to indicate an order is being held before it sent to the
trading service.
, NUM_ORDER_STATUS_CODES
};
```

// Order Status Strings

// Note: These strings are used when saving order status codes to a file.

```
const char* const SCT_OSS_UNSPECIFIED = "Unspecified";
const char* const SCT_OSS_FILLED = "Filled";
const char* const SCT_OSS_CANCELED = "Canceled";
const char* const SCT_OSS_ERROR = "Error";
const char* const SCT_OSS_ORDERSENT = "Order Sent";
const char* const SCT_OSS_PENDINGOPEN = "Pending Open";
const char* const SCT_OSS_HELD = "Held";
const char* const SCT_OSS_OPEN = "Open";
const char* const SCT_OSS_PENDINGCANCEL = "Pending Cancel";
const char* const SCT_OSS_PENDINGMODIFY = "Pending Modify";
const char* const SCT_OSS_PENDING_CHILD_CLIENT = "Pending Child-Client";
const char* const SCT_OSS_PENDING_CHILD_SERVER = "Pending Child-Server";
const char* const SCT_OSS_PENDING_CANCEL_FOR_REPLACE = "Cancel/Replace";
```

```
/*=====
This function returns true when the given order status code is considered
to be in a "working" order state. This means the order can potentially
be filled or be canceled by the user, or canceled by the trading service.
```

Pending child orders are also considered working in this case.

```
-----*/
inline bool IsWorkingOrderStatus(SCOrderStatusCodeEnum OrderStatusCode)
```

```
{
    switch (OrderStatusCode)
    {
        case SCT_OSC_ORDERSENT:
        case SCT_OSC_PENDINGOPEN:
        case SCT_OSC_HELD:
        case SCT_OSC_OPEN:
        case SCT_OSC_PENDINGCANCEL:
        case SCT_OSC_PENDINGMODIFY:
        case SCT_OSC_PENDING_CHILD_CLIENT:
        case SCT_OSC_PENDING_CHILD_SERVER:
        case SCT_OSC_PENDING_CANCEL_FOR_REPLACE:
            return true;

        default:
            return false;
    }
}
```

```
/*=====
This function returns true when the given order status code is considered
to be in a "working" order state. This means the order can potentially
be filled or be canceled by the user, or canceled by the trading service.
```

A pending child order whether on the client or server is not considered working.

```
-----*/
inline bool IsWorkingOrderStatusIgnorePendingChildren(SCOrderStatusCodeEnum OrderStatusCode)
{
```

```

switch (OrderStatusCode)
{
    case SCT_OSC_ORDERSENT:
    case SCT_OSC_PENDINGOPEN:
    case SCT_OSC_HELD:
    case SCT_OSC_OPEN:
    case SCT_OSC_PENDINGCANCEL:
    case SCT_OSC_PENDINGMODIFY:
    case SCT_OSC_PENDING_CANCEL_FOR_REPLACE:
        return true;

    default:
        return false;
}
}

/*=====*/
inline bool IsPendingModifyOrCancelOrderStatus(const SCOrderStatusCodeEnum OrderStatusCode)
{
    switch (OrderStatusCode)
    {
        case SCT_OSC_PENDINGCANCEL:
        case SCT_OSC_PENDINGMODIFY:
        case SCT_OSC_PENDING_CANCEL_FOR_REPLACE:
            return true;

        default:
            return false;
    }
}

/*=====*/
inline bool IsPendingModifyOrderStatus(const SCOrderStatusCodeEnum OrderStatusCode)
{
    return (OrderStatusCode == SCT_OSC_PENDINGMODIFY || OrderStatusCode ==
SCT_OSC_PENDING_CANCEL_FOR_REPLACE);
}

/*=====*/
inline bool IsPendingCancelOrderStatus(const SCOrderStatusCodeEnum OrderStatusCode)
{
    return (OrderStatusCode == SCT_OSC_PENDINGCANCEL);
}

/*=====*/
inline const char* OrderStatusToString(SCOrderStatusCodeEnum OrderStatusCode, bool Abbreviated=false)
{
    switch (OrderStatusCode)
    {
        case SCT_OSC_UNSPECIFIED:
            return SCT_OSS_UNSPECIFIED;

        case SCT_OSC_FILLED:
            return SCT_OSS_FILLED;

        case SCT_OSC_CANCELED:
            return SCT_OSS_CANCELED;

        case SCT_OSC_ERROR:
            return SCT_OSS_ERROR;

        case SCT_OSC_ORDERSENT:
            return SCT_OSS_ORDERSENT;
    }
}

```



```

case SCT_OSC_PENDINGOPEN:
return SCT_OSS_PENDINGOPEN;

case SCT_OSC_HELD:
return SCT_OSS_HELD;

case SCT_OSC_OPEN:
return SCT_OSS_OPEN;

case SCT_OSC_PENDINGCANCEL:
return SCT_OSS_PENDINGCANCEL;

case SCT_OSC_PENDINGMODIFY:
return SCT_OSS_PENDINGMODIFY;

case SCT_OSC_PENDING_CHILD_CLIENT:
    if (Abbreviated)
        return "Child-Client";
    else
        return SCT_OSS_PENDING_CHILD_CLIENT;

case SCT_OSC_PENDING_CHILD_SERVER:
    if (Abbreviated)
        return "Child-Server";
    else
        return SCT_OSS_PENDING_CHILD_SERVER;

case SCT_OSC_PENDING_CANCEL_FOR_REPLACE:
return SCT_OSS_PENDING_CANCEL_FOR_REPLACE;

default:
return "";
}
}

/*=====*/
inline SCOrderStatusCodeEnum OrderStatusStringToEnum(const char* OrderStatusString)
{
    if (_stricmp(OrderStatusString, SCT_OSS_FILLED) == 0)
        return SCT_OSC_FILLED;
    else if (_stricmp(OrderStatusString, SCT_OSS_CANCELED) == 0)
        return SCT_OSC_CANCELED;
    else if (_stricmp(OrderStatusString, SCT_OSS_ERROR) == 0)
        return SCT_OSC_ERROR;
    else if (_stricmp(OrderStatusString, SCT_OSS_ORDERSENT) == 0)
        return SCT_OSC_ORDERSENT;
    else if (_stricmp(OrderStatusString, SCT_OSS_PENDINGOPEN) == 0)
        return SCT_OSC_PENDINGOPEN;
    else if (_stricmp(OrderStatusString, SCT_OSS_HELD) == 0)
        return SCT_OSC_HELD;
    else if (_stricmp(OrderStatusString, SCT_OSS_OPEN) == 0)
        return SCT_OSC_OPEN;
    else if (_stricmp(OrderStatusString, SCT_OSS_PENDINGCANCEL) == 0)
        return SCT_OSC_PENDINGCANCEL;
    else if (_stricmp(OrderStatusString, SCT_OSS_PENDINGMODIFY) == 0)
        return SCT_OSC_PENDINGMODIFY;
    else if (_stricmp(OrderStatusString, SCT_OSS_PENDING_CHILD_CLIENT) == 0)
        return SCT_OSC_PENDING_CHILD_CLIENT;
    else if (_stricmp(OrderStatusString, SCT_OSS_PENDING_CHILD_SERVER) == 0)
        return SCT_OSC_PENDING_CHILD_SERVER;
    else if (_stricmp(OrderStatusString, SCT_OSS_PENDING_CANCEL_FOR_REPLACE) == 0)
        return SCT_OSC_PENDING_CANCEL_FOR_REPLACE;
    else
        return SCT_OSC_UNSPECIFIED;
}

```

// Note: These enumeration values are stored in configuration files, so they  
// must not change.

```
enum BuySellEnum
{ BSE_UNDEFINED = 0
, BSE_BUY = 1
, BSE_SELL = 2
};
```

/\*=====\*/

```
inline const char* BuySellToString(BuySellEnum BuySell)
```

```
{
    switch (BuySell)
    {
        case BSE_BUY:
            return "Buy";

        case BSE_SELL:
            return "Sell";

        default:
            return "";
    }
}
```

/\*=====\*/

```
inline char BuySellToChar(BuySellEnum BuySell)
```

```
{
    switch (BuySell)
    {
        case BSE_BUY:
            return 'B';

        case BSE_SELL:
            return 'S';

        default:
            return 0;
    }
}
```

/\*=====\*/

```
inline BuySellEnum BuySellInverse(BuySellEnum BuySell)
```

```
{
    switch (BuySell)
    {
        case BSE_BUY:
            return BSE_SELL;

        case BSE_SELL:
            return BSE_BUY;

        default:
            return BSE_UNDEFINED;
    }
}
```

// Note: These enumeration values are stored in configuration files, so they  
// must not change.

```
enum OpenCloseEnum
{ OCE_UNDEFINED = 0
, OCE_OPEN = 1
, OCE_CLOSE = 2
};
```

```

/*=====*/
inline const char* OpenCloseToString(OpenCloseEnum OpenClose)
{
    switch (OpenClose)
    {
        case OCE_OPEN:
            return "Open";

        case OCE_CLOSE:
            return "Close";

        default:
            return "";
    }
}

/*=====*/
inline OpenCloseEnum OpenCloseInverse(OpenCloseEnum OpenClose)
{
    switch (OpenClose)
    {
        case OCE_OPEN:
            return OCE_CLOSE;

        case OCE_CLOSE:
            return OCE_OPEN;

        default:
            return OCE_UNDEFINED;
    }
}

enum MoveToBreakEvenActionTypeEnum : int
{ MOVETO_BE_ACTION_TYPE_NONE = 0
, MOVETO_BE_ACTION_TYPE_OFFSET_TRIGGERED = 1
, MOVETO_BE_ACTION_TYPE_OCO_GROUP_TRIGGERED = 2
, MOVETO_BE_ACTION_TYPE_TRAIL_TO_BREAKEVEN = 3
, MOVETO_BE_ACTION_TYPE_OFFSET_TRIGGERED_TRAIL_TO_BREAKEVEN = 4
, MOVETO_BE_ACTION_TYPE_COUNT = 5
};

//OCO group constants
const int OCO_GROUP_1 = 0;
const int OCO_GROUP_2 = 1;
const int OCO_GROUP_3 = 2;
const int OCO_GROUP_4 = 3;
const int OCO_GROUP_5 = 4;
const int OCO_GROUP_6 = 5;
const int OCO_GROUP_7 = 6;
const int OCO_GROUP_8 = 7;

enum ReplayStatus {REPLAY_STOPPED = 0, REPLAY_RUNNING = 1, REPLAY_PAUSED = 2};

//Date-time to string constants
static const int FLAG_DT_YEAR = 0x0001;
static const int FLAG_DT_MONTH = 0x0002;
static const int FLAG_DT_DAY = 0x0004;
static const int FLAG_DT_HOUR = 0x0008;
static const int FLAG_DT_MINUTE = 0x0010;
static const int FLAG_DT_SECOND = 0x0020;
static const int FLAG_DT_PLUS_WITH_TIME = 0x0040;
static const int FLAG_DT_NO_ZERO_PADDING_FOR_DATE = 0x0080;
static const int FLAG_DT_HIDE_SECONDS_IF_ZERO = 0x0100;
static const int FLAG_DT_NO_HOUR_PADDING = 0x0200;

```

```

static const int FLAG_DT_MILLISECOND           = 0x0400;
static const int FLAG_DT_COMPACT_DATE          = 0x0800; // DDMonYY
static const int FLAG_DT_MILLISECOND_OPT       = 0x1000;
static const int FLAG_DT_WEEKDAY               = 0x2000;
static const int FLAG_DT_SINGLE_SPACE          = 0x4000;

static const int FLAG_DT_DATE_TIME_AS_TIME_DURATION_FORMAT = 0x8000;
static const int FLAG_DT_MICROSECOND = 0x10000;
static const int FLAG_DT_COMPLETE_DATE = FLAG_DT_YEAR | FLAG_DT_MONTH | FLAG_DT_DAY;
static const int FLAG_DT_COMPLETE_TIME = FLAG_DT_HOUR | FLAG_DT_MINUTE | FLAG_DT_SECOND;
static const int FLAG_DT_COMPLETE_TIME_MS = FLAG_DT_HOUR | FLAG_DT_MINUTE | FLAG_DT_SECOND |
FLAG_DT_MILLISECOND;
static const int FLAG_DT_COMPLETE_TIME_US = FLAG_DT_HOUR | FLAG_DT_MINUTE | FLAG_DT_SECOND |
FLAG_DT_MICROSECOND;
static const int FLAG_DT_COMPLETE_DATETIME = FLAG_DT_YEAR | FLAG_DT_MONTH | FLAG_DT_DAY |
FLAG_DT_HOUR | FLAG_DT_MINUTE | FLAG_DT_SECOND;
static const int FLAG_DT_COMPLETE_DATETIME_MS = FLAG_DT_YEAR | FLAG_DT_MONTH | FLAG_DT_DAY |
FLAG_DT_HOUR | FLAG_DT_MINUTE | FLAG_DT_SECOND | FLAG_DT_MILLISECOND;
static const int FLAG_DT_COMPLETE_DATETIME_US = FLAG_DT_YEAR | FLAG_DT_MONTH | FLAG_DT_DAY |
FLAG_DT_HOUR | FLAG_DT_MINUTE | FLAG_DT_SECOND | FLAG_DT_MICROSECOND;

```

```

enum HistoricalChartBarPeriodEnum : int32_t
{
    HISTORICAL_CHART_PERIOD_DAYS = 1
    , HISTORICAL_CHART_PERIOD_WEEKLY = 2
    , HISTORICAL_CHART_PERIOD_MONTHLY = 3
    , HISTORICAL_CHART_PERIOD_QUARTERLY = 4
    , HISTORICAL_CHART_PERIOD_YEARLY = 5
};

```

```

enum ContinuousFuturesContractOptionsEnum
{
    CFCO_NONE = 0
    , CFCO_DATE_RULE_ROLLOVER = 1
    , CFCO_VOLUME_BASED_ROLLOVER = 2
    , CFCO_DATE_RULE_ROLLOVER_BACK_ADJUSTED = 3
    , CFCO_VOLUME_BASED_ROLLOVER_BACK_ADJUSTED = 4
    , CFCO_FORWARD_CURVE_CURRENT_DAY = 5
    , CFCO_ROLLOVER_EACH_YEAR_SAME_MONTH = 6
    , CFCO_FORWARD_CURVE_STUDY_SUPPORT = 7
};

```

```

enum IntradayDataFileRecordingStateEnum
{
    IDFRS_NOT_RECORDING_DATA
    , IDFRS_HISTORICAL_DATA_DOWNLOAD_PENDING
    , IDFRS_DOWNLOADING_HISTORICAL_DATA
    , IDFRS_RECEIVING_REALTIME_DATA
    , IDFRS_FINISHED_RECEIVING_DATA
};

```

```

enum MarketDepthUpdateOperationEnum
{
    MDUO_INSERT = 0
    , MDUO_UPDATE = 1
    , MDUO_DELETE = 2
    , MDUO_INSERT_OR_UPDATE = 3
};

```

```

enum ChartWindowStateEnum
{
    CWS_MINIMIZE = 1
    , CWS_RESTORE = 2
    , CWS_MAXIMIZE = 3
};

```

```

enum PeakValleyTypeEnum
{
    PEAKVALLEYTYPE_NONE = 0
    , PEAKVALLEYTYPE_PEAK = 1
    , PEAKVALLEYTYPE_VALLEY = 2
};

namespace n_ACSIL
{
    enum DOMColumnTypeEnum
    {
        DOM_COLUMN_PRICE = 1
        , DOM_COLUMN_BUY_ORDER = 2
        , DOM_COLUMN_SELL_ORDER = 3
        , DOM_COLUMN_BID_SIZE = 4
        , DOM_COLUMN_ASK_SIZE = 5
        , DOM_COLUMN_COMBINED_BID_ASK_SIZE = 6
        , DOM_COLUMN_BID_SIZE_BUY = 7
        , DOM_COLUMN_ASK_SIZE_SELL = 8
        , DOM_COLUMN_LAST_SIZE = 9
        , DOM_COLUMN_CUMULATIVE_LAST_SIZE = 10
        , DOM_COLUMN_RECENT_BID_VOLUME = 11
        , DOM_COLUMN_RECENT_ASK_VOLUME = 12
        , DOM_COLUMN_CURRENT_TRADED_BID_VOLUME = 13
        , DOM_COLUMN_CURRENT_TRADED_ASK_VOLUME = 14
        , DOM_COLUMN_CURRENT_TRADED_TOTAL_VOLUME = 15
        , DOM_COLUMN_BID_MARKET_DEPTH_PULLING_STACKING = 16
        , DOM_COLUMN_ASK_MARKET_DEPTH_PULLING_STACKING = 17
        , DOM_COLUMN_COMBINED_BID_ASK_MARKET_DEPTH_PULLING_STACKING = 18
        , DOM_COLUMN_PROFIT_AND_LOSS = 19
        , DOM_COLUMN_SUBGRAPH_LABELS = 20
        , DOM_COLUMN_GENERAL_PURPOSE_1 = 21
        , DOM_COLUMN_GENERAL_PURPOSE_2 = 22
        , DOM_COLUMN_BID_NUM_ORDERS = 23
        , DOM_COLUMN_ASK_NUM_ORDERS = 24
        , DOM_COLUMN_BID_MARKET_ORDERS = 25
        , DOM_COLUMN_ASK_MARKET_ORDERS = 26
        , DOM_COLUMN_COMBINED_BID_ASK_MARKET_ORDERS = 27
        , DOM_COLUMN_COMBINED_BID_ASK_NUM_ORDERS = 28
    };

```

// These are intended to match exactly the DTC Protocol Security Types values.

```

enum DTCSecurityTypeEnum : int32_t
{
    SECURITY_TYPE_UNSET = 0
    , SECURITY_TYPE_FUTURES = 1
    , SECURITY_TYPE_STOCK = 2
    , SECURITY_TYPE_FOREX = 3 // Cryptocurrencies also go into this category
    , SECURITY_TYPE_INDEX = 4
    , SECURITY_TYPE_FUTURES_STRATEGY = 5
    , SECURITY_TYPE_FUTURES_OPTION = 7
    , SECURITY_TYPE_STOCK_OPTION = 6
    , SECURITY_TYPE_INDEX_OPTION = 8
    , SECURITY_TYPE_BOND = 9
    , SECURITY_TYPE_MUTUAL_FUND = 10
};

```

```

enum OpenFileModeEnum
{
    FILE_MODE_CREATE_AND_OPEN_FOR_READ_WRITE = 1
    , FILE_MODE_OPEN_EXISTING_FOR_SEQUENTIAL_READING = 2
    , FILE_MODE_OPEN_TO_APPEND = 3
    , FILE_MODE_OPEN_TO_REWRITE_FROM_START = 4
};

```

```

enum TradeStatisticsTypeEnum
{
    STATS_TYPE_ALL_TRADES = 0

```

```
, STATS_TYPE_LONG_TRADES = 1
, STATS_TYPE_SHORT_TRADES = 2
, STATS_TYPE_DAILY_ALL_TRADES = 3
```

```
};
```

```
enum ChartReplayModeEnum : int32_t
```

```
{
    REPLAY_MODE_UNSET = 0
, REPLAY_MODE_STANDARD = 1
, REPLAY_MODE_ACCURATE_TRADING_SYSTEM_BACK_TEST = 2
, REPLAY_MODE_CALCULATE_AT_EVERY_TICK = 3
, REPLAY_MODE_CALCULATE_SAME_AS_REAL_TIME = 4
};
```

```
enum ChartsToReplayEnum : int32_t
```

```
{
    CHARTS_TO_REPLAY_UNSET = -1
, CHARTS_TO_REPLAY_SINGLE_CHART = 0
, CHARTS_TO_REPLAY_ALL_CHARTS_IN_CHARTBOOK = 1
, CHARTS_TO_REPLAY_CHARTS_WITH_SAME_LINK_NUMBER = 2
};
```

```
enum GraphicsSettingsEnum : int32_t
```

```
{
    GRAPHICS_SETTING_CHART_TEXT = 0
, GRAPHICS_SETTING_CHART_VALUES_SCALE_TEXT = 1
, GRAPHICS_SETTING_CHART_TIME_SCALE_TEXT = 2
, GRAPHICS_SETTING_CHART_BACKGROUND = 3
, GRAPHICS_SETTING_CHART_GRID = 4
, GRAPHICS_SETTING_CHART_GRID_SECONDARY = 5
, GRAPHICS_SETTING_CHART_ROLLOVER_DATE_LINE = 6
, GRAPHICS_SETTING_CHART_SCALE_BORDER = 7
, GRAPHICS_SETTING_LAST_TRADE_PRICE_BOX = 8
, GRAPHICS_SETTING_LAST_TRADE_PRICE_BOX_FOREGROUND = 9
, GRAPHICS_SETTING_LAST_TRADE_PRICE_BOX_OUTLINE = 10
, GRAPHICS_SETTING_BAR_HIGH_LOW_DOWN = 11
, GRAPHICS_SETTING_BAR_HIGH_LOW_UP = 12
, GRAPHICS_SETTING_BAR_OPEN = 13
, GRAPHICS_SETTING_BAR_CLOSE = 14
, GRAPHICS_SETTING_LINE_ON_CLOSE_MOUNTAIN = 15
, GRAPHICS_SETTING_CANDLESTICK_UP_OUTLINE = 16
, GRAPHICS_SETTING_CANDLESTICK_UP_FILL = 17
, GRAPHICS_SETTING_CANDLESTICK_DOWN_OUTLINE = 18
, GRAPHICS_SETTING_CANDLESTICK_DOWN_FILL = 19
, GRAPHICS_SETTING_VALUE_CHANGE_UP = 20
, GRAPHICS_SETTING_VALUE_CHANGE_DOWN = 21
, GRAPHICS_SETTING_BID_ASK_AVG_LINE = 22
, GRAPHICS_SETTING_SCROLL_END = 23
, GRAPHICS_SETTING_SCROLL_NOT_END = 24
, GRAPHICS_SETTING_ASK_MARKET_DEPTH_LINE = 25
, GRAPHICS_SETTING_ASK_MARKET_DEPTH_LINE_END = 26
, GRAPHICS_SETTING_BID_MARKET_DEPTH_LINE = 27
, GRAPHICS_SETTING_BID_MARKET_DEPTH_LINE_END = 28
, GRAPHICS_SETTING_ALERT_HIGHLIGHT = 29
, GRAPHICS_SETTING_TIMES_AND_SALES_BID_TRADES_BACKGROUND = 30
, GRAPHICS_SETTING_TIMES_AND_SALES_BID_TRADES_TEXT = 31
, GRAPHICS_SETTING_TIMES_AND_SALES_BELOW_BID_TRADES_BACKGROUND = 32
, GRAPHICS_SETTING_TIMES_AND_SALES_BELOW_BID_TRADES_TEXT = 33
, GRAPHICS_SETTING_TIMES_AND_SALES_ASK_TRADES_BACKGROUND = 34
, GRAPHICS_SETTING_TIMES_AND_SALES_ASK_TRADES_TEXT = 35
, GRAPHICS_SETTING_TIMES_AND_SALES_ABOVE_ASK_TRADES_BACKGROUND = 36
, GRAPHICS_SETTING_TIMES_AND_SALES_ABOVE_ASK_TRADES_TEXT = 37
, GRAPHICS_SETTING_TIMES_AND_SALES_BID_AND_ASK_RECORDS_TRADES_BACKGROUND = 38
, GRAPHICS_SETTING_TIMES_AND_SALES_BID_AND_ASK_RECORDS_TRADES_TEXT = 39
};
```

, GRAPHICS\_SETTING\_TIMES\_AND\_SALES\_GRID = 40  
, GRAPHICS\_SETTING\_TIMES\_AND\_SALES\_BID\_HIGHLIGHT = 41  
, GRAPHICS\_SETTING\_TIMES\_AND\_SALES\_BID\_HIGHLIGHT\_TEXT = 42  
, GRAPHICS\_SETTING\_TIMES\_AND\_SALES\_ASK\_HIGHLIGHT = 43  
, GRAPHICS\_SETTING\_TIMES\_AND\_SALES\_ASK\_HIGHLIGHT\_TEXT = 44  
, GRAPHICS\_SETTING\_CHART\_VALUES\_COLOR\_OF\_THE\_PRICE\_AND\_TIME\_BOXES = 45  
, GRAPHICS\_SETTING\_CHART\_VALUES\_CROSSHAIR = 46  
, GRAPHICS\_SETTING\_COLUMN\_SEPARATOR\_LINES = 47  
, GRAPHICS\_SETTING\_BUY\_COLUMN\_BACK = 48  
, GRAPHICS\_SETTING\_BUY\_COLUMN\_LINE = 49  
, GRAPHICS\_SETTING\_SELL\_COLUMN\_LINE = 50  
, GRAPHICS\_SETTING\_SELL\_COLUMN\_BACK = 51  
, GRAPHICS\_SETTING\_BID\_COLUMN\_BACK = 52  
, GRAPHICS\_SETTING\_ASK\_COLUMN\_BACK = 53  
, GRAPHICS\_SETTING\_ASK\_BACKGROUND = 54  
, GRAPHICS\_SETTING\_BID\_BACKGROUND = 55  
, GRAPHICS\_SETTING\_ASK\_DEPTH\_BACK = 56  
, GRAPHICS\_SETTING\_ASK\_DEPTH = 57  
, GRAPHICS\_SETTING\_BID\_DEPTH\_BACK = 58  
, GRAPHICS\_SETTING\_BID\_DEPTH = 59  
, GRAPHICS\_SETTING\_ASK\_MARKET\_ORDER\_BACK = 60  
, GRAPHICS\_SETTING\_ASK\_MARKET\_ORDER = 61  
, GRAPHICS\_SETTING\_BID\_MARKET\_ORDER\_BACK = 62  
, GRAPHICS\_SETTING\_BID\_MARKET\_ORDER = 63  
, GRAPHICS\_SETTING\_BEST\_BID\_ASK\_BOX\_BACK = 64  
, GRAPHICS\_SETTING\_BUY\_ORDER\_QUANTITY\_TEXT = 65  
, GRAPHICS\_SETTING\_SELL\_ORDER\_QUANTITY\_TEXT = 66  
, GRAPHICS\_SETTING\_DAILY\_HIGH\_LINE = 67  
, GRAPHICS\_SETTING\_DAILY\_LOW\_LINE = 68  
, GRAPHICS\_SETTING\_RECENTER\_LINE = 69  
, GRAPHICS\_SETTING\_PNL\_PLUS = 70  
, GRAPHICS\_SETTING\_PNL\_MINUS = 71  
, GRAPHICS\_SETTING\_VALUES\_SCALE\_TEXT = 72  
, GRAPHICS\_SETTING\_LAST\_AT\_BID\_TEXT = 73  
, GRAPHICS\_SETTING\_LAST\_AT\_ASK\_TEXT = 74  
, GRAPHICS\_SETTING\_LAST\_PRICE\_BACKGROUND\_BID\_TRADE = 75  
, GRAPHICS\_SETTING\_LAST\_PRICE\_BACKGROUND\_ASK\_TRADE = 76  
, GRAPHICS\_SETTING\_CUMULATIVE\_LAST\_SIZE\_BID\_TRADE = 77  
, GRAPHICS\_SETTING\_CUMULATIVE\_LAST\_SIZE\_ASK\_TRADE = 78  
, GRAPHICS\_SETTING\_CUMULATIVE\_LAST\_SIZE\_BACKGROUND\_BID\_TRADE = 79  
, GRAPHICS\_SETTING\_CUMULATIVE\_LAST\_SIZE\_BACKGROUND\_ASK\_TRADE = 80  
, GRAPHICS\_SETTING\_CUMULATIVE\_LAST\_SIZE\_BID\_TRADE\_SAME\_PRICE\_AND\_SIDE\_BACKGROUND =  
81 , GRAPHICS\_SETTING\_CUMULATIVE\_LAST\_SIZE\_ASK\_TRADE\_SAME\_PRICE\_AND\_SIDE\_BACKGROUND =  
82 , GRAPHICS\_SETTING\_CUMULATIVE\_LAST\_SIZE\_BID\_TRADE\_SAME\_PRICE\_AND\_SIDE\_TEXT = 83  
, GRAPHICS\_SETTING\_CUMULATIVE\_LAST\_SIZE\_ASK\_TRADE\_SAME\_PRICE\_AND\_SIDE\_TEXT = 84  
, GRAPHICS\_SETTING\_RECENT\_BID\_VOLUME = 85  
, GRAPHICS\_SETTING\_RECENT\_ASK\_VOLUME = 86  
, GRAPHICS\_SETTING\_LAST\_TRADED\_RECENT\_BID\_VOLUME\_BACK = 87  
, GRAPHICS\_SETTING\_LAST\_TRADED\_RECENT\_ASK\_VOLUME\_BACK = 88  
, GRAPHICS\_SETTING\_LAST\_TRADED\_RECENT\_BID\_VOLUME\_TEXT = 89  
, GRAPHICS\_SETTING\_LAST\_TRADED\_RECENT\_ASK\_VOLUME\_TEXT = 90  
, GRAPHICS\_SETTING\_TRADED\_TOTAL\_VOLUME = 91  
, GRAPHICS\_SETTING\_TRADED\_BID\_VOLUME = 92  
, GRAPHICS\_SETTING\_TRADED\_ASK\_VOLUME = 93  
, GRAPHICS\_SETTING\_BID\_MARKET\_DEPTH\_PULLING\_STACKING = 94  
, GRAPHICS\_SETTING\_BID\_MARKET\_DEPTH\_PULLING\_STACKING\_NEGATIVE = 95  
, GRAPHICS\_SETTING\_ASK\_MARKET\_DEPTH\_PULLING\_STACKING = 96  
, GRAPHICS\_SETTING\_ASK\_MARKET\_DEPTH\_PULLING\_STACKING\_NEGATIVE = 97  
, GRAPHICS\_SETTING\_BID\_VOLUME\_BAR = 98  
, GRAPHICS\_SETTING\_ASK\_VOLUME\_BAR = 99  
, GRAPHICS\_SETTING\_VOLUME\_BARS = 100  
, GRAPHICS\_SETTING\_TOTAL\_PERCENTAGE\_TEXT = 101  
, GRAPHICS\_SETTING\_RECENT\_BID\_VOLUME\_VOLUME\_BAR = 102



```

, GRAPHICS_SETTING_RECENT_BID_VOLUME_BACKGROUND = 103
, GRAPHICS_SETTING_RECENT_ASK_VOLUME_VOLUME_BAR = 104
, GRAPHICS_SETTING_RECENT_ASK_VOLUME_BACKGROUND = 105
, GRAPHICS_SETTING_CURRENT_TRADED_BID_VOLUME_VOLUME_BAR = 106
, GRAPHICS_SETTING_CURRENT_TRADED_BID_VOLUME_BACKGROUND = 107
, GRAPHICS_SETTING_CURRENT_TRADED_ASK_VOLUME_VOLUME_BAR = 108
, GRAPHICS_SETTING_CURRENT_TRADED_ASK_VOLUME_BACKGROUND = 109
, GRAPHICS_SETTING_CURRENT_TRADED_TOTAL_VOLUME_VOLUME_BAR = 110
, GRAPHICS_SETTING_CURRENT_TRADED_TOTAL_VOLUME_BACKGROUND = 111
, GRAPHICS_SETTING_BID_MARKET_DEPTH_PULLING_STACKING_VOLUME_BAR = 112
, GRAPHICS_SETTING_BID_MARKET_DEPTH_PULLING_STACKING_BACKGROUND = 113
, GRAPHICS_SETTING_ASK_MARKET_DEPTH_PULLING_STACKING_VOLUME_BAR = 114
, GRAPHICS_SETTING_ASK_MARKET_DEPTH_PULLING_STACKING_BACKGROUND = 115
, GRAPHICS_SETTING_RECENT_BID_BEST_BID_BACKGROUND = 116
, GRAPHICS_SETTING_RECENT_ASK_BEST_ASK_BACKGROUND = 117
, GRAPHICS_SETTING_RECENT_BID_SAME_PRICE_AND_SIDE_BACKGROUND = 118
, GRAPHICS_SETTING_RECENT_ASK_SAME_PRICE_AND_SIDE_BACKGROUND = 119
, GRAPHICS_SETTING_RECENT_BID_SAME_PRICE_AND_SIDE_TEXT = 120
, GRAPHICS_SETTING_RECENT_ASK_SAME_PRICE_AND_SIDE_TEXT = 121
, GRAPHICS_SETTING_BID_ASK_BAR_ASK_LARGE = 122
, GRAPHICS_SETTING_BID_ASK_BAR_ASK_MEDIUM_LARGE = 123
, GRAPHICS_SETTING_BID_ASK_BAR_ASK_LIGHT_MEDIUM = 124
, GRAPHICS_SETTING_BID_ASK_BAR_ASK_LIGHT = 125
, GRAPHICS_SETTING_BID_ASK_BAR_BID_LARGE = 126
, GRAPHICS_SETTING_BID_ASK_BAR_BID_MEDIUM_LARGE = 127
, GRAPHICS_SETTING_BID_ASK_BAR_BID_LIGHT_MEDIUM = 128
, GRAPHICS_SETTING_BID_ASK_BAR_BID_LIGHT = 129
, GRAPHICS_SETTING_CHART_TRADE_WINDOW_BACKGROUND_SIM = 130
, GRAPHICS_SETTING_CHART_TRADE_MODE_BOX = 131
, GRAPHICS_SETTING_CHART_TRADE_MODE_BOX_SIM = 132
, GRAPHICS_SETTING_CHART_TRADING_ORDERS_BEING_INPUTTED = 133
, GRAPHICS_SETTING_CHART_TRADING_ORDER_LINE_BUY_LIMIT = 134
, GRAPHICS_SETTING_CHART_TRADING_ORDER_LINE_SELL_LIMIT = 135
, GRAPHICS_SETTING_CHART_CHART_TRADING_ORDER_LINE_BUY_STOP = 136
, GRAPHICS_SETTING_CHART_CHART_TRADING_ORDER_LINE_SELL_STOP = 137
, GRAPHICS_SETTING_CHART_CHART_TRADING_ORDER_LINE_BUY_LIMIT_CHILD = 138
, GRAPHICS_SETTING_CHART_CHART_TRADING_ORDER_LINE_SELL_LIMIT_CHILD = 139
, GRAPHICS_SETTING_CHART_CHART_TRADING_ORDER_LINE_BUY_STOP_CHILD = 140
, GRAPHICS_SETTING_CHART_CHART_TRADING_ORDER_LINE_SELL_STOP_CHILD = 141
, GRAPHICS_SETTING_CHART_CHART_TRADING_POSITION_LINE_LONG_PROFIT = 142
, GRAPHICS_SETTING_CHART_CHART_TRADING_POSITION_LINE_SHORT_LOSS = 143
, GRAPHICS_SETTING_CHART_TRADE_WINDOW_BACKGROUND = 144
};
}

```

```

enum IntradayFileLockActionEnum : uint16_t
{
    IFLA_LOCK_READ_HOLD = 1
, IFLA_NO_CHANGE = 2
, IFLA_RELEASE_AFTER_READ = 3
, IFLA_LOCK_READ_RELEASE = 4
};

```

```

const int CUSTOM_CHART_BAR_RELOAD_FLAG_KEY = 1048576;

```

```

// Summary Sub-trade Indicator

```

```

enum UnbundledTradeIndicatorEnum : int16_t
{
    UNBUNDLED_TRADE_NONE = 0,
    FIRST_SUB_TRADE_OF_UNBUNDLED_TRADE = 1,
    LAST_SUB_TRADE_OF_UNBUNDLED_TRADE = 2
};

```

```

enum SymbolDataValuesEnum : int32_t

```

```

{
    SYMBOL_DATA_UNSET = 0,
    SYMBOL_DATA_DAILY_OPEN = 1,
    SYMBOL_DATA_DAILY_HIGH = 2,
    SYMBOL_DATA_DAILY_LOW = 3,
    SYMBOL_DATA_DAILY_NUMBER_OF_TRADES = 4,
    SYMBOL_DATA_LAST_TRADE_PRICE = 5,
    SYMBOL_DATA_LAST_TRADE_VOLUME = 6,
    SYMBOL_DATA_ASK_QUANTITY = 7,
    SYMBOL_DATA_BID_QUANTITY = 8,
    SYMBOL_DATA_BID_PRICE = 9,
    SYMBOL_DATA_ASK_PRICE = 10,
    SYMBOL_DATA_CURRENCY_VALUE_PER_TICK = 11,
    SYMBOL_DATA_SETTLEMENT_PRICE = 12,
    SYMBOL_DATA_OPEN_INTEREST = 13,
    SYMBOL_DATA_DAILY_VOLUME = 14,
    SYMBOL_DATA_SHARES_OUTSTANDING = 15,
    SYMBOL_DATA_EARNINGS_PER_SHARE = 16,
    SYMBOL_DATA_TICK_DIRECTION = 17,
    SYMBOL_DATA_LAST_TRADE_AT_SAME_PRICE = 18,
    SYMBOL_DATA_STRIKE_PRICE = 19,
    SYMBOL_DATA_SELL_ROLLOVER_INTEREST = 20,
    SYMBOL_DATA_PRICE_FORMAT = 21,
    SYMBOL_DATA_BUY_ROLLOVER_INTEREST = 22,
    SYMBOL_DATA_TRADE_INDICATOR = 23,
    SYMBOL_DATA_LAST_TRADE_AT_BID_ASK = 24,
    SYMBOL_DATA_VOLUME_VALUE_FORMAT = 25,
    SYMBOL_DATA_TICK_SIZE = 26,
    SYMBOL_DATA_LAST_TRADE_DATE_TIME = 27,
    SYMBOL_DATA_ACCUMULATED_LAST_TRADE_VOLUME = 28,
    SYMBOL_DATA_LAST_TRADING_DATE_FOR_FUTURES = 29,
    SYMBOL_DATA_TRADING_DAY_DATE = 30,
    SYMBOL_DATA_LAST_MARKET_DEPTH_UPDATE_DATE_TIME = 31,
    SYMBOL_DATA_DISPLAY_PRICE_MULTIPLIER = 32,
    SYMBOL_DATA_SETTLEMENT_PRICE_DATE = 33,
    SYMBOL_DATA_DAILY_OPEN_PRICE_DATE = 34,
    SYMBOL_DATA_DAILY_HIGH_PRICE_DATE = 35,
    SYMBOL_DATA_DAILY_LOW_PRICE_DATE = 36,
    SYMBOL_DATA_DAILY_VOLUME_DATE = 37,
    SYMBOL_DATA_NUMBER_OF_TRADES_AT_CURRENT_PRICE = 38,
    SYMBOL_DATA_TRADING_STATUS = 39
};

```

```

typedef double t_MarketDataQuantity; //for volume and other quantity values related to market data.

```

```

struct s_MarketDepthEntry
{
    float Price = 0;
    t_MarketDataQuantity Quantity = 0;
    uint32_t NumOrders = 0;
    float AdjustedPrice = 0;

    s_MarketDepthEntry(int DummyValue = 0)
    {
    }

    void Clear()
    {
        Price = 0;
        Quantity = 0;
        NumOrders = 0;
        AdjustedPrice = 0;
    }

    bool IsEmpty() const

```

```
{
    return (Price == 0 && Quantity == 0);
}

uint32_t GetQuantityAsInt() const
{
    return static_cast<uint32_t>(Quantity);
}

static bool BidDOMSortPred(const s_MarketDepthEntry& First, const s_MarketDepthEntry& Second)
{
    return (First.Price > Second.Price);
}

static bool AskDOMSortPred(const s_MarketDepthEntry& First, const s_MarketDepthEntry& Second)
{
    return (First.Price < Second.Price);
}

typedef bool(*t_SortPred)(const s_MarketDepthEntry& First, const s_MarketDepthEntry& Second);
};

#endif
```