Movicon® 11

MONITORING VISION AND CONTROL

Supervision and control XML-based
from Windows Vista to Windows CE

VBA Language Reference Guide

Version 11.3- Ed. Feb. 2012

Cod. DOCS 11 VBA-E Build 1101

Table Of Contents

1. PREFACE 37

1.1. INTRODUCTION 37

1.1.1. Preface 37

2. GENERAL CONCEPTS 39

2.1. BASIC SCRIPTS IN PROJECTS 39

2.1.1. WinWrap Basic Language 39

2.2. VB.NET AND UNICODE SUPPORT 39

2.2.1. Basic Script Libraries 40

2.2.2. Subs, Functions, Events, Methods and Properties 42

2.2.3. Variables in Basic Scripts 45

2.2.4. Quick Programming 46

2.3. CODE IN BASIC SCRIPTS AND IN SYMBOLS 47

2.3.1. Basic Scripts as Resources 47

2.3.2. Basic Script Expressions in Object properties 49

2.3.3. VBA™ Basic Script in Object and Alarm Code 51

2.4. PUBLIC BASIC SCRIPTS 52

2.5. BASIC SCRIPT EDITOR 53

2.5.1. Edit Menu (Basic Scripts) 54

2.5.2. Debug Menu 56

2.5.3. Basic Script ToolBar 57

2.6. THE SCRIPT EXPLORER WINDOW 58

2.7. BASIC SCRIPT DEBUG 60

2.8. BASIC SCRIPT PROPERTIES 61

2.8.1. Basic Script General Properties 61

2.8.2. Basic Script Mode Properties 61

2.8.3. Basic Script Execution Properties 62

2.8.4. Script Debug Output 63

2.9. ADO IN BASIC SCRIPTS 63

2.10. ACTIVEX/OCX IN BASIC SCRIPTS 64

2.10.1. How to use ActiveX with Licenses 65

2.10.2. Example of using an ActiveX/OCX into a Basic Script 67

3. API BASIC INTERFACE 69

3.1. USING BASIC SCRIPT INTERFACES 69

4. ALARMCMDTARGET 73

GetAlarmThreshold, AlarmCmdTarget Function 73

GetXMLSettings, AlarmCmdTarget Function 73

AlarmOnQualityGood, AlarmCmdTarget Property 74

DeviceName, AlarmCmdTarget Property 74

Enabled, AlarmCmdTarget Property 75

EnableVariableName, AlarmCmdTarget Property 75

Isteresis, AlarmCmdTarget Property 76

Name, AlarmCmdTarget Property 76

ThresholdExclusive, AlarmCmdTarget Property 77

VariableName, AlarmCmdTarget Property 77

5. ALARMTHRESHOLDCMDTARGET 78

AlarmLoading, AlarmThresholdCmdTarget Event 78

AlarmUnloading, AlarmThresholdCmdTarget Event 78

OnAckAlarm, AlarmThresholdCmdTarget Event 78

OnCommentAlarm, AlarmThresholdCmdTarget Event 78

OnHelpAlarm, AlarmThresholdCmdTarget Event 79

OnResetAlarm, AlarmThresholdCmdTarget Event 79

OnSetAlarm, AlarmThresholdCmdTarget Event *79*
ResetAlarm, AlarmThresholdCmdTarget Function *80*
GetAlarmObject, AlarmThresholdCmdTarget Function *80*
GetTotNumAck, AlarmThresholdCmdTarget Function *81*
GetTotNumOn, AlarmThresholdCmdTarget Function *81*
GetTotNumReset, AlarmThresholdCmdTarget Function *82*
GetTransactionID, AlarmThresholdCmdTarget Function *82*
GetUniqueID, AlarmThresholdCmdTarget Function *83*
GetXMLSettings, AlarmThresholdCmdTarget Function *83*
AckAlarm, AlarmThresholdCmdTarget Function *84*
AlarmArea, AlarmThresholdCmdTarget Property *84*
Attachment, AlarmThresholdCmdTarget Property *85*
BackColor, AlarmThresholdCmdTarget Property *85*
Beep, AlarmThresholdCmdTarget Property *85*
BlinkBackColor, AlarmThresholdCmdTarget Property *86*
BlinkOnNewAlarm, AlarmThresholdCmdTarget Property *86*
BlinkTextColor, AlarmThresholdCmdTarget Property *87*
CommandList..., AlarmThresholdCmdTarget Property *87*
CommentOnAck, AlarmThresholdCmdTarget Property *88*
Condition, AlarmThresholdCmdTarget Property *88*
DateTimeACK, AlarmThresholdCmdTarget Property *89*
DateTimeACKMs, AlarmThresholdCmdTarget Property *89*
DateTimeFromTimeStamp, AlarmThresholdCmdTarget Property *90*
DateTimeOFF, AlarmThresholdCmdTarget Property *90*
DateTimeOFFMs, AlarmThresholdCmdTarget Property *91*
DateTimeON, AlarmThresholdCmdTarget Property *91*
DateTimeOnMs, AlarmThresholdCmdTarget Property *92*
DateTimeRESET, AlarmThresholdCmdTarget Property *92*
DateTimeRESETMs, AlarmThresholdCmdTarget Property *93*
DurationFormat, AlarmThresholdCmdTarget Property *93*
Help, AlarmThresholdCmdTarget Property *94*
LastComment, AlarmThresholdCmdTarget Property *94*
LastTotalTimeOn, AlarmThresholdCmdTarget Property *95*
Log, AlarmThresholdCmdTarget Property *95*
Name, AlarmThresholdCmdTarget Property *96*
PlaysoundContinuously, AlarmThresholdCmdTarget Property *96*
Print, AlarmThresholdCmdTarget Property *97*
ReadAccessLevel, AlarmThresholdCmdTarget Property *97*
Recipient, AlarmThresholdCmdTarget Property *98*
RepeatSpeechEverySec, AlarmThresholdCmdTarget Property *98*
SecDelay, AlarmThresholdCmdTarget Property *99*
SendFaxEnabledACK, AlarmThresholdCmdTarget Property *99*
SendFaxEnabledOFF, AlarmThresholdCmdTarget Property *100*
SendFaxEnabledON, AlarmThresholdCmdTarget Property *100*
SendFaxEnabledRESET, AlarmThresholdCmdTarget Property *101*
SendMailEnabledACK, AlarmThresholdCmdTarget Property *101*
SendMailEnabledOFF, AlarmThresholdCmdTarget Property *102*
SendMailEnabledON, AlarmThresholdCmdTarget Property *102*
SendMailEnabledRESET, AlarmThresholdCmdTarget Property *103*
SendSMSEnabledACK, AlarmThresholdCmdTarget Property *103*
SendSMSEnabledOFF, AlarmThresholdCmdTarget Property *104*
SendSMSEnabledON, AlarmThresholdCmdTarget Property *104*
SendSMSEnabledRESET, AlarmThresholdCmdTarget Property *105*
SendVoiceEnabledACK, AlarmThresholdCmdTarget Property *105*
SendVoiceEnabledOFF, AlarmThresholdCmdTarget Property *106*
SendVoiceEnabledON, AlarmThresholdCmdTarget Property *106*
SendVoiceEnabledRESET, AlarmThresholdCmdTarget Property *107*
Severity, AlarmThresholdCmdTarget Property *107*
SpeechEnabled, AlarmThresholdCmdTarget Property *107*
SpeechEnableVariable, AlarmThresholdCmdTarget Property *108*
Status, AlarmThresholdCmdTarget Property *108*
StatusVariable, AlarmThresholdCmdTarget Property *109*
SupportAcknowledge, AlarmThresholdCmdTarget Property *109*
SupportReset, AlarmThresholdCmdTarget Property *110*
TextColor, AlarmThresholdCmdTarget Property *110*
SupportResetWithConditionOn, AlarmThresholdCmdTarget Property *111*
Text, AlarmThresholdCmdTarget Property *111*
Threshold, AlarmThresholdCmdTarget Property *112*

ThresholdLow, AlarmThresholdCmdTarget Property *112*
TotalTimeOn, AlarmThresholdCmdTarget Property *113*
VariableSeverity, AlarmThresholdCmdTarget Property *113*
VariableThreshold, AlarmThresholdCmdTarget Property *114*
VariableThresholdLow, AlarmThresholdCmdTarget Property *114*
WriteAccessLevel, AlarmThresholdCmdTarget Property *115*

6. ALARMWNDCMDTARGET 116

OnAckAll, AlarmWndCmdTarget Event *116*
OnAckSel, AlarmWndCmdTarget Event *116*
OnGetHistory, AlarmWndCmdTarget Event *116*
OnHelp, AlarmWndCmdTarget Event *116*
OnCommentSel, AlarmWndCmdTarget Event *117*
OnInsertOrUpdateAlarm, AlarmWndCmdTarget Event *117*
OnOPCAEEvent, AlarmWndCmdTarget Event *117*
OnResetAll, AlarmWndCmdTarget Event *118*
OnResetSelSel, AlarmWndCmdTarget Event *118*
OnToggleSound, AlarmWndCmdTarget Event *118*
AckSelectedAlarms, AlarmWndCmdTarget Function *118*
EditLayout, AlarmWndCmdTarget Function *119*
EditCopy, AlarmWndCmdTarget Function *119*
GetNumTotalAlarms, AlarmWndCmdTarget Function *120*
GetSelectedAlarm, AlarmWndCmdTarget Function *120*
GetSelHistory, AlarmWndCmdTarget Function *121*
LoadExtSettings, AlarmWndCmdTarget Function *121*
RecalcLayout, AlarmWndCmdTarget Function *122*
Refresh, AlarmWndCmdTarget Function *122*
ResetSelectedAlarms, AlarmWndCmdTarget Function *123*
SaveExtSettings, AlarmWndCmdTarget Function *123*
SelectAll, AlarmWndCmdTarget Function *124*
AckAllBtnText, AlarmWndCmdTarget Property *124*
AckSelBtnText, AlarmWndCmdTarget Property *125*
AlarmFilter, AlarmWndCmdTarget Property *125*
AlarmFilterMask, AlarmWndCmdTarget Property *125*
AlarmFilterSeverity, AlarmWndCmdTarget Property *126*
AlarmFilterSeverityCondition, AlarmWndCmdTarget Property *127*
AreaFilter, AlarmWndCmdTarget Property *127*
AutoLayout, AlarmWndCmdTarget Property *127*
Autoscroll, AlarmWndCmdTarget Property *128*
BlinkTime, AlarmWndCmdTarget Property *128*
ButtonPos, AlarmWndCmdTarget Property *129*
ButtonSize, AlarmWndCmdTarget Property *129*
Clickable, AlarmWndCmdTarget Property *130*
ExtSettingsFile, AlarmWndCmdTarget Property *130*
FormatDateTime, AlarmWndCmdTarget Property *131*
FormatDuration, AlarmWndCmdTarget Property *131*
GetHistoryBtnTex, AlarmWndCmdTarget Property *132*
GraphicButtons, AlarmWndCmdTarget Property *132*
HasSpin, AlarmWndCmdTarget Property *133*
HelpBtnText, AlarmWndCmdTarget Property *133*
HisLogBackColor, AlarmWndCmdTarget Property *134*
HisLogTextColor, AlarmWndCmdTarget Property *134*
HorizontalSpin, AlarmWndCmdTarget Property *135*
IncludeMilliseconds, AlarmWndCmdTarget Property *135*
MaxOPCAEEvents, AlarmWndCmdTarget Property *136*
NetworkBackupServerName, AlarmWndCmdTarget Property *137*
NetworkServer, AlarmWndCmdTarget Property *137*
OPCAEServer, AlarmWndCmdTarget Property *138*
RefreshTimePoll, AlarmWndCmdTarget Property *138*
ResetAllSelBtnText, AlarmWndCmdTarget Property *138*
ResetSelBtnText, AlarmWndCmdTarget Property *139*
ScrollTime, AlarmWndCmdTarget Property *139*
ShowAckAllBtn, AlarmWndCmdTarget Property *140*
ShowAckSelBtn, AlarmWndCmdTarget Property *140*
ShowDateTime, AlarmWndCmdTarget Property *141*
ShowGetHistoryBtn, AlarmWndCmdTarget Property *141*
ShowHelpBtn, AlarmWndCmdTarget Property *142*

ShowHigherSeverity, AlarmWndCmdTarget Property *142*
ShowResetAllBtn, AlarmWndCmdTarget Property *143*
ShowResetSelBtn, AlarmWndCmdTarget Property *144*
ShowSoundOnBtn, AlarmWndCmdTarget Property *144*
SpinSize, AlarmWndCmdTarget Property *145*
SoundOnBtnText, AlarmWndCmdTarget Property *145*
SubItemAck, AlarmWndCmdTarget Property *146*
SubItemAckPos, AlarmWndCmdTarget Property *146*
SubItemAckWidth, AlarmWndCmdTarget Property *147*
SubItemCondition, AlarmWndCmdTarget Property *147*
SubItemConditionPos, AlarmWndCmdTarget Property *148*
SubItemConditionWidth, AlarmWndCmdTarget Property *148*
SubItemDuration, AlarmWndCmdTarget Property *149*
SubItemDurationPos, AlarmWndCmdTarget Property *149*
SubItemDurationWidth, AlarmWndCmdTarget Property *150*
SubItemImage, AlarmWndCmdTarget Property *150*
SubItemImagePos, AlarmWndCmdTarget Property *151*
SubItemImageWidth, AlarmWndCmdTarget Property *151*
SubItemOff, AlarmWndCmdTarget Property *152*
SubItemOffPos, AlarmWndCmdTarget Property *152*
SubItemOffWidth, AlarmWndCmdTarget Property *153*
SubItemOn, AlarmWndCmdTarget Property *153*
SubItemOnPos, AlarmWndCmdTarget Property *154*
SubItemOnWidth, AlarmWndCmdTarget Property *154*
SubItemReset, AlarmWndCmdTarget Property *155*
SubItemResetPos, AlarmWndCmdTarget Property *155*
SubItemResetWidth, AlarmWndCmdTarget Property *156*
SubItemSeverity, AlarmWndCmdTarget Property *156*
SubItemSeverityPos, AlarmWndCmdTarget Property *157*
SubItemSeverityWidth, AlarmWndCmdTarget Property *157*
SubItemStatus, AlarmWndCmdTarget Property *158*
SubItemStatusPos, AlarmWndCmdTarget Property *158*
SubItemStatusWidth, AlarmWndCmdTarget Property *159*
SubItemText, AlarmWndCmdTarget Property *159*
SubItemTextPos, AlarmWndCmdTarget Property *160*
SubItemTextWidth, AlarmWndCmdTarget Property *160*
SubItemTotalNumAck, AlarmWndCmdTarget Property *161*
SubItemTotalNumAckPos, AlarmWndCmdTarget Property *161*
SubItemTotalNumAckWidth, AlarmWndCmdTarget Property *162*
SubItemTotalNumOn, AlarmWndCmdTarget Property *162*
SubItemTotalNumOnPos, AlarmWndCmdTarget Property *163*
SubItemTotalNumOnWidth, AlarmWndCmdTarget Property *163*
SubItemTotalNumReset, AlarmWndCmdTarget Property *164*
SubItemTotalNumResetPos, AlarmWndCmdTarget Property *164*
SubItemTotalNumResetWidth, AlarmWndCmdTarget Property *165*
SubItemTotalTimeOn, AlarmWndCmdTarget Property *165*
SubItemTotalTimeOnPos, AlarmWndCmdTarget Property *166*
SubItemTotalTimeOnWidth, AlarmWndCmdTarget Property *166*

7. BUTTONCMDTARGET 167

GetCommandsInterfaceOnPressed, ButtonCmdTarget Function *167*
GetCommandsInterfaceOnRelease, ButtonCmdTarget Function *167*
GetCommandsInterfaceWhileDown, ButtonCmdTarget Function *168*
GetShorcutText, ButtonCmdTarget Function *168*
AsciiKeyShortcut, ButtonCmdTarget Property *169*
Border, ButtonCmdTarget Property *169*
ButtonStyle, ButtonCmdTarget Property *170*
Clickable, ButtonCmdTarget Property *171*
CommandStateVariable, ButtonCmdTarget Property *172*
CommandType, ButtonCmdTarget Property *172*
DisableCommandsOnCheckedState, ButtonCmdTarget Property *173*
EnableShortcut, ButtonCmdTarget Property *173*
ExecuteCommandsOnMouseMove, ButtonCmdTarget Property *174*
ImageBtnChecked, ButtonCmdTarget Property *174*
ImageBtnDisabled, ButtonCmdTarget Property *175*
ImageBtnPressed, ButtonCmdTarget Property *175*
ImageBtnReleased, ButtonCmdTarget Property *176*

ImpulsiveTime, ButtonCmdTarget Property *176*
OverlapImageText, ButtonCmdTarget Property *177*
RadioBtnNumOptions, ButtonCmdTarget Property *178*
RadioCheckBtnSize, ButtonCmdTarget Property *178*
Round3DStyle, ButtonCmdTarget Property *179*
ShowShortcut, ButtonCmdTarget Property *179*
TriStateCentralZero, ButtonCmdTarget Property *180*
VirtualKeyShortcut, ButtonCmdTarget Property *180*

8. CHARTWNDCMDTARGET 182

OnErrorRecordset, ChartWndCmdTarget Event *182*
OnRecordsetMoveNext, ChartWndCmdTarget Event *182*
OnRecordsetQueryEnd, ChartWndCmdTarget Event *182*
OnRecordsetQueryStart, ChartWndCmdTarget Event *182*
GetChartInterface, ChartWndCmdTarget Function *183*
LoadExtSettings, ChartWndCmdTarget Function *183*
SaveExtSettings, ChartWndCmdTarget Function *184*
RecalcLayout, ChartWndCmdTarget Function *184*
AddStackVariable, ChartWndCmdTarget Property *185*
ArrayType, ChartWndCmdTarget Property *185*
BackupLink, ChartWndCmdTarget Property *186*
Border, ChartWndCmdTarget Property *187*
Clickable, ChartWndCmdTarget Property *187*
DataDefaultQuery, ChartWndCmdTarget Property *188*
DataFilterBy, ChartWndCmdTarget Property *189*
DataSortBy, ChartWndCmdTarget Property *189*
ElevationVariable, ChartWndCmdTarget Property *190*
ExtSettingsFile, ChartWndCmdTarget Property *191*
LinkedDataLogger, ChartWndCmdTarget Property *191*
NetworkBackupServerName, ChartWndCmdTarget Property *192*
NetworkServerName, ChartWndCmdTarget Property *192*
NumSamples, ChartWndCmdTarget Property *193*
RotationVariable, ChartWndCmdTarget Property *193*
Title1, ChartWndCmdTarget Property *194*
Title2, ChartWndCmdTarget Property *194*
Title3, ChartWndCmdTarget Property *195*
Title4, ChartWndCmdTarget Property *195*
Title5, ChartWndCmdTarget Property *196*
Title6, ChartWndCmdTarget Property *196*
Title7, ChartWndCmdTarget Property *197*
Title8, ChartWndCmdTarget Property *197*
Variable1, ChartWndCmdTarget Property *198*
Variable2, ChartWndCmdTarget Property *199*
Variable3, ChartWndCmdTarget Property *199*
Variable4, ChartWndCmdTarget Property *200*
Variable5, ChartWndCmdTarget Property *200*
Variable6, ChartWndCmdTarget Property *201*
Variable7, ChartWndCmdTarget Property *201*
Variable8, ChartWndCmdTarget Property *202*

9. CLIENTRULESINTERFACE 203

ClientTimeout, ClientRulesInterface Property *203*
DefaultClientUser, ClientRulesInterface Property *203*
Name, ClientRulesInterface Property *204*
PingTime, ClientRulesInterface Property *204*
Priority, ClientRulesInterface Property *205*
Protocol, ClientRulesInterface Property *205*
RasStation, ClientRulesInterface Property *206*
UseRASStation, ClientRulesInterface Property *207*

10. COMMANDALARMCMDTARGET 208

ConvertPeriodNumToString, CommandAlarmCmdTarget Function *208*
GetCommandBaseInterface, CommandAlarmCmdTarget Function *209*
Action, CommandAlarmCmdTarget Property *210*
AreaFilter,CommandAlarmCmdTarget Property *211*

PrintSettingsLandscape, CommandAlarmCmdTarget Property *212*
PrintSettingsPageHeight, CommandAlarmCmdTarget Property *214*
PrintSettingsPageWidth, CommandAlarmCmdTarget Property *215*
PrintSettingsPortSettings, CommandAlarmCmdTarget Property *216*
PrintSettingsPrinterName, CommandAlarmCmdTarget Property *217*
PrintSettingsPrinterPort, CommandAlarmCmdTarget Property *219*
PrintSettingsShowPrintDialog, CommandAlarmCmdTarget Property *220*
Recipient, CommandAlarmCmdTarget Property *222*
StatisticRptFile, CommandAlarmCmdTarget Property *223*
StatisticRptReferenceDate, CommandAlarmCmdTarget Property *224*
StatisticRptReferenceDuration, CommandAlarmCmdTarget Property *225*
StatisticRptReferencePeriod, CommandAlarmCmdTarget Property *227*
StatisticRptShowToolbar, CommandAlarmCmdTarget Property *228*
StatisticRptShowTree, CommandAlarmCmdTarget Property *229*
TextualRptBottomMargin, CommandAlarmCmdTarget Property *231*
TextualRptLeftMargin, CommandAlarmCmdTarget Property *232*
TextualRptMaxPages, CommandAlarmCmdTarget Property *233*
TextualRptOutputFile, CommandAlarmCmdTarget Property *234*
TextualRptRightMargin, CommandAlarmCmdTarget Property *236*
TextualRptSQLQuery, CommandAlarmCmdTarget Property *237*
TextualRptTemplateFile, CommandAlarmCmdTarget Property *238*
TextualRptTopMargin, CommandAlarmCmdTarget Property *239*

11. COMMANDBASECMDTARGET 241

Type, CommandBaseCmdTarget Property *241*
XmlSettings, CommandBaseCmdTarget Property *242*

12. COMMANDEVENTCMDTARGET 244

Event, CommandEventCmdTarget Property *244*
GetCommandBaseInterface, CommandEventCmdTarget Function *245*

13. COMMANDHELPCMDTARGET 247

GetCommandBaseInterface, CommandHelpCmdTarget Function *247*
Action, CommandHelpCmdTarget Property *248*
Topic, CommandHelpCmdTarget Property *249*

14. COMMANDLANGUAGECMDTARGET 251

Language, CommandLanguageCmdTarget Property *251*
GetCommandBaseInterface, CommandLanguageCmdTarget Function *252*

15. COMMANDSLISTCMDTARGET 254

AddToHead, CommandsListCmdTarget Function *254*
AddToTail, CommandsListCmdTarget Function *255*
DiscardChanges, CommandsListCmdTarget Function *256*
GetCommandInterfaceAtPos, CommandsListCmdTarget Function *257*
GetCommandTypeAtPos, CommandsListCmdTarget Function *258*
GetTotNumCommands, CommandsListCmdTarget Function *259*
InsertAfter, CommandsListCmdTarget Function *259*
InsertBefore, CommandsListCmdTarget Function *260*
MoveToHead, CommandsListCmdTarget Function *261*
MoveToTail, CommandsListCmdTarget Function *262*
RemoveAll, CommandsListCmdTarget Function *263*
RemoveAtPos, CommandsListCmdTarget Function *264*
RemoveFromHead, CommandsListCmdTarget Function *264*
RemoveFromTail, CommandsListCmdTarget Function *265*
SaveChanges, CommandsListCmdTarget Function *266*
SetAtPos, CommandsListCmdTarget Function *267*
SwapCommands, CommandsListCmdTarget Function *268*

16. COMMANDMENUCMDTARGET 269

Menu, CommandMenuCmdTarget Property *269*
XPos, CommandMenuCmdTarget Property *270*
YPos, CommandMenuCmdTarget Property *271*

GetCommandBaseInterface, CommandMenuCmdTarget Function *272*

17. COMMANDREPORTCMDTARGET 274

GetCommandBaseInterface, CommandReportCmdTarget Function *274*
Action, CommandReportCmdTarget Property *275*
DLR, CommandReportCmdTarget Property *276*
EmbeddedReportName, CommandReportCmdTarget Property *277*
Height, CommandReportCmdTarget Property *279*
Landscape, CommandReportCmdTarget Property *280*
PageHeight, CommandReportCmdTarget Property *281*
PageWidth, CommandReportCmdTarget Property *282*
PortSettings, CommandReportCmdTarget Property *283*
PrinterName, CommandReportCmdTarget Property *285*
PrinterPort, CommandReportCmdTarget Property *286*
RecipeCSVSeparator, CommandReportCmdTarget Property *288*
Recipient,CommandReportCmdTarget Property *289*
ReportExportFormat, CommandReportCmdTarget Property *290*
ReportReferencePeriod, CommandReportCmdTarget Property *292*
ReportShowFilterByDate, CommandReportCmdTarget Property *293*
ReportShowToolbar, CommandReportCmdTarget Property *294*
ReportShowTree, CommandReportCmdTarget Property *295*
ShowPrintDialog, CommandReportCmdTarget Property *297*
TextualRptBottomMargin, CommandReportCmdTarget Property *298*
TextualRptLeftMargin, CommandReportCmdTarget Property *299*
TextualRptMaxPages, CommandReportCmdTarget Property *300*
TextualRptOutputFile, CommandReportCmdTarget Property *302*
TextualRptRightMargin, CommandReportCmdTarget Property *303*
TextualRptSQLQuery, CommandReportCmdTarget Property *304*
TextualRptTemplateFile, CommandReportCmdTarget Property *305*
TextualRptTopMargin, CommandReportCmdTarget Property *307*
XPos, CommandReportCmdTarget Property *308*
YPos, CommandReportCmdTarget Property *309*

18. COMMANDSCRIPTCMDTARGET 311

GetCommandBaseInterface, CommandScriptCmdTarget Function *311*
Action, CommandScriptCmdTarget Property *312*
MoreInstanceAllowed, CommandScriptCmdTarget Property *313*
Parameters, CommandScriptCmdTarget Property *314*
Script, CommandScriptCmdTarget Property *315*
SynchroTimeout, CommandScriptCmdTarget Property *317*
SynopticName, CommandSynopticCmdTarget Property *318*
Width, CommandReportCmdTarget Property *319*

19. COMMANDSYNOPTICCMDTARGET 321

GetCommandBaseInterface, CommandSynopticCmdTarget Function *321*
Action, CommandSynopticCmdTarget Property *322*
Height, CommandSynopticCmdTarget Property *323*
KeeproportionsOnPrint, CommandSynopticCmdTarget Property *324*
Monitor, CommandSynopticCmdTarget Property *325*
ParameterFile, CommandSynopticCmdTarget Property *327*
PrintBottomMargin, CommandSynopticCmdTarget Property *328*
PrintLeftMargin, CommandSynopticCmdTarget Property *329*
PrintPageHeight, CommandSynopticCmdTarget Property *330*
PrintPageWidth, CommandSynopticCmdTarget Property *331*
PrintRightMargin, CommandSynopticCmdTarget Property *333*
PrintTopMargin, CommandSynopticCmdTarget Property *334*
ResizeableBorder, CommandSynopticCmdTarget Property *335*
ShowBorder, CommandSynopticCmdTarget Property *336*
ShowCaption, CommandSynopticCmdTarget Property *337*
ShowMaximizedBtn, CommandSynopticCmdTarget Property *339*
ShowMinimizedBtn, CommandSynopticCmdTarget Property *340*
ShowSystemMenu, CommandSynopticCmdTarget Property *341*
Width, CommandSynopticCmdTarget Property *342*
XPos, CommandSynopticCmdTarget Property *344*
YPos, CommandSynopticCmdTarget Property *345*

20. COMMANDSYSTEMCMDTARGET 347

GetCommandBaseInterface, CommandSystemCmdTarget Function 347
Action, CommandSystemCmdTarget Property 348
CommandLine, CommandSystemCmdTarget Property.....349
Timeout, CommandSystemCmdTarget Property 350
WorkingPath, CommandSystemCmdTarget Property 351

21. COMMANDUSERSCMDTARGET 354

GetCommandBaseInterface, CommandUsersCmdTarget Function 354
Action, CommandUsersCmdTarget Property 355
Level, CommandUsersCmdTarget Property.....356

22. COMMANDVARIABLECMDTARGET.....358

GetCommandBaseInterface, CommandVariableCmdTarget Function 358
Action, CommandVariableCmdTarget Property.....359
MaxChar, CommandVariableCmdTarget Property.....360
MaxValue, CommandVariableCmdTarget Property 361
MinValue, CommandVariableCmdTarget Property 363
MoveToVariable, CommandVariableCmdTarget Property 364
PasswordStyle, CommandVariableCmdTarget Property 365
StrobeTime, CommandVariableCmdTarget Property.....366
Value, CommandVariableCmdTarget Property.....367
Variable, CommandVariableCmdTarget Property.....368

23. DBVARIABLECMDTARGET.....371

CreateNewVar, DBVariableCmdTarget Function 371
GetDynamicVariable, DBVariableCmdTarget Function.....372
GetNumDynTag, DBVariableCmdTarget Function.....372
GetNumStructDefinitions, DBVariableCmdTarget Function.....373
GetNumVariables, DBVariableCmdTarget Function.....373
GetRealTimeDBADOConn, DBVariableCmdTarget Function.....374
GetTraceDBADOConn, DBVariableCmdTarget Function 375
GetTraceDBDSNConnectionString, DBVariableCmdTarget Function.....375
GetVariableAddressInfo, DBVariableCmdTarget Function.....376
GetVariableDescription, DBVariableCmdTarget Function 377
GetVariableObject, DBVariableCmdTarget Function 377
GetVariableSize, DBVariableCmdTarget Function.....378
GetVariableValue, DBVariableCmdTarget Function 378
GetXMLSettings, DBVariableCmdTarget Function 379
IsFirstDBInstance, DBVariableCmdTarget Function.....379
IsValidVariable, DBVariableCmdTarget Function 380
PurgeDynTag, DBVariableCmdTarget Function.....380
QualityOf, DBVariableCmdTarget Function 381
SetVariableValue, DBVariableCmdTarget Function.....381
VariableInUse, DBVariableCmdTarget Function 382
EnableInUseVarMng, DBVariableCmdTarget Property.....382
EnableNTSecurityOPCServerTag, DBVariableCmdTarget Property 383
OPCServerEnableAEAck, DBVariableCmdTarget Property 383
OPCServerMinImpersonationLevel, DBVariableCmdTarget Property 384
OPCServerShutdownClientsTimeout, DBVariableCmdTarget Property.....384
PurgeDynTagTimer, DBVariableCmdTarget Property 385
TraceDBChangerColName, DBVariableCmdTarget Property 385
TraceDBDefVarCharPrecision, DBVariableCmdTarget Property 386
TraceDBDsn, DBVariableCmdTarget Property.....386
TraceDBLocalTimeColName, DBVariableCmdTarget Property 387
TraceDBMaxCacheBeforeFlush, DBVariableCmdTarget Property.....388
TraceDBMaxError, DBVariableCmdTarget Property.....388
TraceDBMaxNumberTrans, DBVariableCmdTarget Property.....388
TraceDBMSecColName, DBVariableCmdTarget Property.....389
TraceDBQualityColName, DBVariableCmdTarget Property 389
TraceDBRecycleDBConnection, DBVariableCmdTarget Property 390
TraceDBTimeColName, DBVariableCmdTarget Property 390
TraceDBTimeStampColName, DBVariableCmdTarget Property.....391
TraceDBUser, DBVariableCmdTarget Property.....392
TraceDBUserColName, DBVariableCmdTarget Property 392

TraceDBValueAfterColName, DBVariableCmdTarget Property 393
TraceDBValueBeforeColName, DBVariableCmdTarget Property 393
TraceDBValueColName, DBVariableCmdTarget Property 394
TraceDBVarDescNameColName, DBVariableCmdTarget Property 394
TraceDBVarGroupNameColName, DBVariableCmdTarget Property 395
TraceDBVarNameColName, DBVariableCmdTarget Property 395
TraceUseIMDB,DBVariableCmdTarget Property 396
UseSharedDynTag, DBVariableCmdTarget Property 396

24. DBVAROBJCMDTARGET 398

GetAccessLevelReadMask, DBVarObjCmdTarget Function 398
GetAccessLevelWriteMask, DBVarObjCmdTarget Function 398
GetAddress, DBVarObjCmdTarget Function 399
GetAlarmListName, DBVarObjCmdTarget Function 399
GetAlarmListNum, DBVarObjCmdTarget Function 400
GetAlarmObject, DBVarObjCmdTarget Function 400
GetAreaType, DBVarObjCmdTarget Function 401
GetBitNumber, DBVarObjCmdTarget Function 402
GetDataLoggerListNum, DBVarObjCmdTarget Function 402
GetDataLoggerName, DBVarObjCmdTarget Function 403
Description, DBVarObjCmdTarget Function 403
GetEventListName, DBVarObjCmdTarget Function 404
GetEventListNum, DBVarObjCmdTarget Function 404
GetEventObject, DBVarObjCmdTarget Function 405
GetInitialTimeInUse, DBVarObjCmdTarget Function 406
GetInUseCount, DBVarObjCmdTarget Function 406
GetInUseObjectAt, DBVarObjCmdTarget Function 407
GetInUseObjectNameAt, DBVarObjCmdTarget Function 407
GetLastTimeInUse, DBVarObjCmdTarget Function 408
GetMemberObjectFromIndex, DBVarObjCmdTarget Function 408
GetMemberObjectFromName, DBVarObjCmdTarget Function 409
GetName, DBVarObjCmdTarget Function 410
GetNumObjectsInHeap, DBVarObjCmdTarget Function 410
GetStructName, DBVarObjCmdTarget Function 411
GetStructParentObject, DBVarObjCmdTarget Function 411
GetTimeNotInUse, DBVarObjCmdTarget Function 412
GetTimeStamp, DBVarObjCmdTarget Function 412
GetType, DBVarObjCmdTarget Function 413
GetXMLSettings, DBVarObjCmdTarget Function 413
IsOPCServerEnabled, DBVarObjCmdTarget Function 414
IsOPCServerOnRequest, DBVarObjCmdTarget Function 414
IsShared, DBVarObjCmdTarget Function 415
IsValid, DBVarObjCmdTarget Function 415
ResetStatisticData, DBVarObjCmdTarget Function 416
SetOPCServerEnabled, DBVarObjCmdTarget Function 416
SetTimeStamp, DBVarObjCmdTarget Function 417
AviFileProp, DBVarObjCmdTarget Property 418
BGColorProp, DBVarObjCmdTarget Property 418
BlinkProp, DBVarObjCmdTarget Property 418
BmpFileProp, DBVarObjCmdTarget Property 419
CloseBitString, DBVarObjCmdTarget Property 419
DynamicSettings, DBVarObjCmdTarget Property 420
EnableFactor, DBVarObjCmdTarget Property 420
EnableNetworkServer, DBVarObjCmdTarget Property 421
EnableScalingFactor, DBVarObjCmdTarget Property 421
EngineeringUnit, DBVarObjCmdTarget Property 422
FactorGain, DBVarObjCmdTarget Property 422
FactorOffset, DBVarObjCmdTarget Property 423
FGColorProp, DBVarObjCmdTarget Property 423
Group, DBVarObjCmdTarget Property 424
HtmlFileProp DBVarObjCmdTarget Property 424
InUse, DBVarObjCmdTarget Property 425
InverseFactor, DBVarObjCmdTarget Property 425
InverseScaling, DBVarObjCmdTarget Property 426
LastChangeComment, DBVarObjCmdTarget Property 426
MapRealTimeODBCUpdateQuality, DBVarObjCmdTarget Property 427
MapRealTimeToDB, DBVarObjCmdTarget Property 427

MapRealTimeToDBMode, DBVarObjCmdTarget Property *428*
MapRealTimeToDBRefreshTime, DBVarObjCmdTarget Property *428*
NetworkClientEnable, DBVarObjCmdTarget Property *429*
NetworkClientMode, DBVarObjCmdTarget Property *429*
NetworkClientServerName, DBVarObjCmdTarget Property *430*
NetworkClientUpdateQuality, DBVarObjCmdTarget Property *430*
NetworkServerEnable, DBVarObjCmdTarget Property *431*
NetworkServerIsWritable, DBVarObjCmdTarget Property *431*
OPCGroupName, DBVarObjCmdTarget Property *432*
OPCServerAccessRights,DBVarObjTarget Property *432*
OpenBitString, DBVarObjCmdTarget Property *433*
Quality, DBVarObjCmdTarget Property *433*
ScaleMax, DBVarObjCmdTarget Property *434*
ScaleMin, DBVarObjCmdTarget Property *435*
ScaleRawMax, DBVarObjCmdTarget Property *435*
ScaleRawMin, DBVarObjCmdTarget Property *436*
SndFileProp DBVarObjCmdTarget Property *436*
StatisticData DBVarObjCmdTarget Property *436*
StatisticDataAverage, DBVarObjCmdTarget Property *437*
StatisticDataMaxValue, DBVarObjCmdTarget Property *437*
StatisticDataMinValue, DBVarObjCmdTarget Property *438*
StatisticDataNumSamples, DBVarObjCmdTarget Property *438*
StatisticTotalTimeOn, DBVarObjCmdTarget Property *439*
TraceAddDescCol, DBVarObjCmdTarget Property *439*
TraceAddGroupCol, DBVarObjCmdTarget Property *440*
TraceAddMsgLog, DBVarObjCmdTarget Property *440*
TraceComment, DBVarObjCmdTarget Property *441*
TraceEnable, DBVarObjCmdTarget Property *441*
TraceEnableFromToTime, DBVarObjCmdTarget Property *442*
TraceFromTime, DBVarObjCmdTarget Property *442*
TraceMaxAgeDays, DBVarObjCmdTarget Property *443*
TraceMaxAgeHours, DBVarObjCmdTarget Property *443*
TraceMaxAgeMins, DBVarObjCmdTarget Property *444*
TraceTableName, DBVarObjCmdTarget Property *445*
TraceToTime, DBVarObjCmdTarget Property *445*
Value, DBVarObjCmdTarget Property *446*

25. DISPLAYEDITCMDTARGET 447

GetComboListInterface, DisplayEditCmdTarget Function *447*
IsCombo, DisplayEditCmdTarget Function *447*
IsSpin, DisplayEditCmdTarget Function *448*
LoadExtSettings, DisplayEditCmdTarget Function *448*
RecalcLayout, DisplayEditCmdTarget Function *449*
SaveExtSettings, DisplatEditCmdTarget Function *449*
EditingPassword, DisplayEditCmdTarget Property *450*
FormatData, DisplayEditCmdTarget Property *450*
ExtSettingsFile, DisplayEditCmdTarget Property *451*
FormatVariable, DisplayEditCmdTarget Property *451*
HasSpin, DisplayEditCmdTarget Property *452*
InvertSelection, DisplayEditCmdTarget Property *452*
IsEditable, DisplayEditCmdTarget Property *453*
IsSpinHoriz, DisplayEditCmdTarget Property *453*
PromptPad, DisplayEditCmdTarget Property *453*
ScaleUnit, DisplayEditCmdTarget Property *454*
SpinStep, DisplayEditCmdTarget Property *454*
TimeToWaitToIncrease, DisplayEditCmdTarget Property *455*
ValMax, DisplayEditCmdTarget Property *455*
ValMin, DisplayEditCmdTarget Property *456*
ValueToDisplay, DisplayEditCmdTarget Property *456*
Variable, DisplayEditCmdTarget Property *457*
VariableMax, DisplayEditCmdTarget Property *457*
VariableMin, DisplayEditCmdTarget Property *458*

26. DLRCMDTARGET 459

GetADOConn, DLRCmdTarget Function *459*
GetColumn, DLRCmdTarget Function *460*

GetColumnNameList, DLRCmdTarget Function *460*
GetDNSConnectionString, DLRCmdTarget Function *461*
GetNextTickLocalTime, DLRCmdTarget Function *461*
GetNextTickTime, DLRCmdTarget Function *462*
ActivateVariable, DLRCmdTarget Property *462*
CRWReportFile, DLRCmdTarget Property *463*
DeleteVariable, DLRCmdTarget Property *463*
DSN, DLRCmdTarget Property *464*
DurationDays, DLRCmdTarget Property *464*
DurationHours, DLRCmdTarget Property *465*
DurationMinutes, DLRCmdTarget Property *466*
Enabled, DLRCmdTarget Property *466*
EnableTimeFrom, DLRCmdTarget Property *467*
EnableTimeFromTo, DLRCmdTarget Property *467*
EnableTimeTo, DLRCmdTarget Property *468*
EnableTimeVariable, DLRCmdTarget Property *468*
ExecuteVariable, DLRCmdTarget Property *469*
Filter, DLRCmdTarget Property *469*
FilterVariable, DLRCmdTarget Property *470*
InsertVariable, DLRCmdTarget Property *470*
IsRecipe, DLRCmdTarget Property *471*
LocalTimeColName, DLRCmdTarget Property *471*
MaxCacheBeforeFlush, DLRCmdTarget Property *472*
MaxError, DLRCmdTarget Property *472*
MaxNumberTrans, DLRCmdTarget Property *473*
MoveFirstVariable, DLRCmdTarget Property *473*
MoveLastVariable, DLRCmdTarget Property *474*
MoveNextVariable, DLRCmdTarget Property *474*
MovePrevVariable, DLRCmdTarget Property *475*
MSecColName, DLRCmdTarget Property *476*
Name, DLRCmdTarget Property *476*
PrintVariable, DLRCmdTarget Property *477*
Query, DLRCmdTarget Property *477*
QueryVariable, DLRCmdTarget Property *478*
ReadVariable, DLRCmdTarget Property *478*
ReasonColName, DLRCmdTarget Property *479*
RecipeIndexName, DLRCmdTarget Property *479*
RecordOnChange, DLRCmdTarget Property *480*
RecordOnChangeDeadBand, DLRCmdTarget Property *481*
RecordOnChangeDeadBandPercent, DLRCmdTarget Property *481*
RecordOnChangeEnableDeadBand, DLRCmdTarget Property *482*
RecordOnlyWhenQualityGood, DLRCmdTarget Property *482*
RecordOnTime, DLRCmdTarget Property *483*
RecordOnVariable, DLRCmdTarget Property *483*
RecVariable, DLRCmdTarget Property *484*
RecycleDBConnection, DLRCmdTarget Property *484*
ResetVariable, DLRCmdTarget Property *485*
Sort, DLRCmdTarget Property *485*
SortVariable, DLRCmdTarget Property *486*
StatusVariable, DLRCmdTarget Property *486*
TableName, DLRCmdTarget Property *487*
TimeColName, DLRCmdTarget Property *487*
TimeRecHour, DLRCmdTarget Property *488*
TimeRecMin, DLRCmdTarget Property *488*
TimeRecMSec, DLRCmdTarget Property *489*
TimeRecSec, DLRCmdTarget Property *489*
UseIMDB,DLRCmdTarget property *489*
UserColName, DLRCmdTarget Property *490*
UserName, DLRCmdTarget Property *490*
VarCharsMax, DLRCmdTarget Property *491*

27. DLRCOLUMNCMDTARGET 492

AddNumUpdatesCol, DLRColumnCmdTarget Property *492*
AddQualityColumn, DLRColumnCmdTarget Property *492*
Name, DLRColumnCmdTarget Property *493*
NumUpdatesColumnName, DLRColumnCmdTarget Property *493*
QualityColumnName, DLRColumnCmdTarget Property *494*

RecipeIndex, DLRColumnCmdTarget Property *494*
RecipeTempVariable, DLRColumnCmdTarget Property *495*
RecordType, DLRColumnCmdTarget Property *496*
StatisticAverageValue, DLRColumnCmdTarget Property *497*
StatisticMaxValue, DLRColumnCmdTarget Property *497*
StatisticMinValue, DLRColumnCmdTarget Property *498*
StatisticNumUpdates, DLRColumnCmdTarget Property *498*
Variable, DLRColumnCmdTarget Property *499*

28. DLRWNDCMDTARGET 500

OnFilter, DLRWndCmdTarget Event *500*
OnPrint, DLRWndCmdTarget Event *500*
OnRefresh, DLRWndCmdTarget Event *500*
EditCopy, DLRWndCmdTarget Function *500*
EditLayout, DLRWndCmdTarget Function *501*
LoadExtSettings, DLRWndCmdTarget Function *501*
RecalcLayout, DLRWndCmdTarget Function *502*
Refresh, DLRWndCmdTarget Function *502*
SaveExtSettings, DLRWndCmdTarget Function *503*
AutoLayout, DLRWndCmdTarget Property *503*
ButtonPos, DLRWndCmdTarget Property *504*
ButtonSize, DLRWndCmdTarget Property *504*
Clickable, DLRWndCmdTarget Property *505*
DLR, DLRWndCmdTarget Property *506*
ExtSettingsFile, DLRWndCmdTarget Property *506*
FilterBtnText, DLRWndCmdTarget Property *506*
FilterFromDate, DLRWndCmdTarget Property *507*
FilterToDate, DLRWndCmdTarget Property *508*
FilterUser, DLRWndCmdTarget Property *508*
GraphicButtons, DLRWndCmdTarget Property *509*
IncludeMilliseconds, DLRWndCmdTarget Property *509*
MaxCount, DLRWndCmdTarget Property *510*
NetworkBackupServerName, DLRWndCmdTarget Property *510*
NetworkServerName, DLRWndCmdTarget Property *511*
PrintBtnText, DLRWndCmdTarget Property *511*
Project, DLRWndCmdTarget Property *512*
Query, DLRWndCmdTarget Property *513*
RefreshBtnText, DLRWndCmdTarget Property *513*
ShowFilterBtn, DLRWndCmdTarget Property *514*
ShowPrintBtn, DLRWndCmdTarget Property *514*
ShowRefreshBtn, DLRWndCmdTarget Property *515*
SortBy, DLRWndCmdTarget Property *515*
SubItemReason, DLRWndCmdTarget Property *516*
SubItemReasonPos, DLRWndCmdTarget Property *517*
SubItemReasonWidth, DLRWndCmdTarget Property *517*
SubItemTime, DLRWndCmdTarget Property *518*
SubItemTimePos, DLRWndCmdTarget Property *518*
SubItemTimeWidth, DLRWndCmdTarget Property *519*
SubItemUser, DLRWndCmdTarget Property *519*
SubItemUserPos, DLRWndCmdTarget Property *520*
SubItemUserWidth, DLRWndCmdTarget Property *520*

29. DRAWCMDTARGET 521

Click, Generic Event *521*
DblClick, Generic Event *521*
KeyDown, Generic Event *521*
KeyPress, Generic Event *521*
KeyUp, Generic Event *522*
KillFocus, Generic Event *522*
MouseDown, Generic Event *522*
MouseMove, Generic Event *523*
MouseUp, Generic Event *523*
OnChange, Generic Event *524*
OnChangeExecutionCanceled, Generic Event *524*
OnChangeExecutionToPromoter, Generic Event *524*
OnExecutionPending, Generic Event *525*

OnFireExecution, Generic Event *525*
OnFireSynapse, Generic Event *525*
OnPostPaint, Generic Event *525*
OnPrePaint, Generic Event *526*
OnTextChanged, Generic Event *526*
OnTextChanging, Generic Event *526*
OnTimer, Generic Event *527*
OnToolTip, Generic Event *527*
SetFocus, Generic Event *527*
SymbolLoading, Generic Event *527*
SymbolUnloading, Generic Event *528*
AddPolyPoint, DrawCmdTarget Function *528*
CloseThisSynoptic, DrawCmdTarget Function *528*
ConvertAngleToPoint, DrawCmdTarget Function *529*
ConvertPointToAngle, DrawCmdTarget Function *530*
CursorPosToObjectPos, DrawCmdTarget Function *531*
DeletePolyPoint, DrawCmdTarget Function *531*
EnableVariableEvent, DrawCmdTarget Function *532*
GetActiveXObject, DrawCmdTarget Function *532*
GetAlias, DrawCmdTarget Function *533*
GetAliasListName, DrawCmdTarget Function *533*
GetAliasListValue, DrawCmdTarget Function *534*
GetCommandsInterfaceOnRelease, DrawCmdTarget Function *534*
GetConnectorObjectConnected, DrawCmdTarget Function *535*
GetContainerObject, DrawCmdTarget Function *535*
GetCursorPos, DrawCmdTarget Function *536*
GetCursorPosInObject, DrawCmdTarget Function *536*
GetGaugeObject, DrawCmdTarget Function *537*
GetNumConnectionsOnSynapse, DrawCmdTarget Function *537*
GetNumPolyPoint, DrawCmdTarget Function *538*
GetNumSynapsis, DrawCmdTarget Function *538*
GetObjectConnectedOnSynapse, DrawCmdTarget Function *539*
GetObjectInterface, DrawCmdTarget Function *539*
GetOnScreenPosition, DrawCmdTarget Function *540*
GetPolyPointOnScreenX, DrawCmdTarget Function *541*
GetPolyPointOnScreenY, DrawCmdTarget Function *541*
GetSubGaugeObject, DrawCmdTarget Function *542*
GetSubObject, DrawCmdTarget Function *542*
GetSubTrendObject, DrawCmdTarget Function *543*
GetSynapseName, DrawCmdTarget Function *543*
GetSynapsePoint, DrawCmdTarget Function *544*
GetSynopticObject, DrawCmdTarget Function *544*
GetTrendObject, DrawCmdTarget Function *545*
GetUniqueObjectID, DrawCmdTarget Function *546*
GetXMLSettings, DrawCmdTarget Function *546*
HasSynapsis, DrawCmdTarget Function *546*
InflateObject, DrawCmdTarget Function *547*
IsCursorOnObject, DrawCmdTarget Function *547*
IsGlobalObjectName, DrawCmdTarget Function *548*
IsSynapseConnected, DrawCmdTarget Function *548*
LoadExtSettings, DrawCmdTarget Function *549*
MoveObject, DrawCmdTarget Function *549*
OffsetObject, DrawCmdTarget Function *550*
PolyPointX, DrawCmdTarget Function *550*
PolyPointY, DrawCmdTarget Function *551*
PrintThisSynoptic, DrawCmdTarget Function *551*
Prop, DrawCmdTarget Function *552*
RemoveAlias, DrawCmdTarget Function *553*
RemoveAllAliases, DrawCmdTarget Function *553*
ResetColors, DrawCmdTarget Function *553*
SaveExtSettings, DrawCmdTarget Function *554*
ScaleObject, DrawCmdTarget Function *554*
SetAlias, DrawCmdTarget Function *555*
ShowPropList, DrawCmdTarget Function *556*
SynapseBackColor, DrawCmdTarget Property *556*
SynapsePassExecution, DrawCmdTarget Function *557*
SynapseValueFromID, DrawCmdTarget Property *557*
ZOrderMoveBack, DrawCmdTarget Function *558*

ZOrderMoveForward, DrawCmdTarget Function *558*
ZOrderMoveToBack, DrawCmdTarget Function *558*
ZOrderMoveToFront, DrawCmdTarget Function *559*
AdaptFontSize, DrawCmdTarget Property *559*
AlignFont, DrawCmdTarget Property *560*
AlignFontOffsetX, DrawCmdTarget Property *561*
AlignFontOffsetY, DrawCmdTarget Property *561*
AutoRepeatClick, DrawCmdTarget Property *562*
BackBrushPattern, DrawCmdTarget Property *562*
BackBrushVisible, DrawCmdTarget Property *563*
BackColor, DrawCmdTarget Property *563*
BackColorBrightness, DrawCmdTarget Property *564*
BackColorHue, DrawCmdTarget Property *565*
BackColorSaturation, DrawCmdTarget Property *565*
BitmapAlignment, DrawCmdTarget Property *566*
BitmapID, DrawCmdTarget Property *567*
BitmapOffsetX, DrawCmdTarget Property *567*
BitmapOffsetY, DrawCmdTarget Property *568*
BitmapStretched, DrawCmdTarget Property *568*
BitmapTransparent, DrawCmdTarget Property *569*
BitmapTransparentColor, DrawCmdTarget Property *569*
BitmapKeepAspectRatio, DrawCmdTarget Property *570*
BorderType, DrawCmdTarget Property *570*
CenterRotation, DrawCmdTarget Property *571*
DefStructName, DrawCmdTarget Property *571*
DefStructNameAbsolute, DrawCmdTarget Property *572*
DrawingState, DrawCmdTarget Property *572*
DrawingStateShadow, DrawCmdTarget Property *573*
EdgeColor, DrawCmdTarget Property *574*
EmbeddedSynoptic, DrawCmdTarget Property *574*
EnableExecution, DrawCmdTarget Property *575*
EnableVariable, DrawCmdTarget Property *575*
ExtSettingsFile, DrawCmdTarget Property *576*
FillBrushPattern, DrawCmdTarget Property *576*
FillColor, DrawCmdTarget Property *577*
FillingMode, DrawCmdTarget Property *577*
FillingPercent, DrawCmdTarget Property *578*
Font3D, DrawCmdTarget Property *578*
FontBold, DrawCmdTarget Property *579*
FontCharSet, DrawCmdTarget Property *579*
FontEscapement, DrawCmdTarget Property *580*
FontHeight, DrawCmdTarget Property *581*
FontItalic, DrawCmdTarget Property *581*
FontName, DrawCmdTarget Property *582*
GradientColor, DrawCmdTarget Property *582*
GradientFill, DrawCmdTarget Property *582*
Height, DrawCmdTarget Property *583*
Hilite, DrawCmdTarget Property *584*
LineArrowHeight, DrawCmdTarget Property *584*
LineArrowType, DrawCmdTarget Property *585*
LineEndingX, DrawCmdTarget Property *585*
LineEndingY, DrawCmdTarget Property *586*
LineStartingX, DrawCmdTarget Property *586*
LineStartingY, DrawCmdTarget Property *587*
LinkedTextFormat, DrawCmdTarget Property *587*
LinkedTextFormatVariable, DrawCmdTarget Property *588*
Look3D, DrawCmdTarget Property *588*
Look3DPressed, DrawCmdTarget Property *589*
MetaFile, DrawCmdTarget Property *589*
MouseCapture, DrawCmdTarget Property *590*
ObjectName, DrawCmdTarget Property *590*
ObjectPublicName, DrawCmdTarget Property *590*
PenColorBrightness, DrawCmdTarget Property *591*
PenColorHue, DrawCmdTarget Property *592*
PenColorSaturation, DrawCmdTarget Property *592*
PenStyle, DrawCmdTarget Property *593*
PenVisible, DrawCmdTarget Property *593*
PenWidth, DrawCmdTarget Property *594*

Rotation, DrawCmdTarget Property *594*
Shadow, DrawCmdTarget Property *595*
ShadowColor, DrawCmdTarget Property *595*
ShadowXOffset, DrawCmdTarget Property *596*
ShadowYOffset, DrawCmdTarget Property *596*
ShowFocusRectangle, DrawCmdTarget Property *597*
ShowHiliteRectangle, DrawCmdTarget Property *597*
StatusVariable, DrawCmdTarget Property *598*
SynapseValue, DrawCmdTarget Property *598*
SynapsisVisible, DrawCmdTarget Property *599*
Title, DrawCmdTarget Property *599*
ToolTip, DrawCmdTarget Property *600*
Transparency, DrawCmdTarget Property *600*
VariableBackColor, DrawCmdTarget Property *601*
VariableBitmapList, DrawCmdTarget Property *601*
VariableComposedMovement, DrawCmdTarget Property *601*
VariableEdgeColor, DrawCmdTarget Property *602*
VariableEndingX, DrawCmdTarget Property *602*
VariableEndingY, DrawCmdTarget Property *603*
VariableFillColor, DrawCmdTarget Property *603*
VariableFilling, DrawCmdTarget Property *604*
VariableLinkedText, DrawCmdTarget Property *604*
VariableMoveX, DrawCmdTarget Property *605*
VariableMoveY, DrawCmdTarget Property *605*
VariableRotation, DrawCmdTarget Property *605*
VariableScaling, DrawCmdTarget Property *606*
VariableStartingX, DrawCmdTarget Property *606*
VariableStartingY, DrawCmdTarget Property *607*
VariableVisible, DrawCmdTarget Property *607*
VariableXRotationCenter, DrawCmdTarget Property *608*
VariableYRotationCenter, DrawCmdTarget Property *608*
Visible, DrawCmdTarget Property *609*
Width, DrawCmdTarget Property *609*
Xpos, DrawCmdTarget Property *610*
XRotationCenter, DrawCmdTarget Property *610*
Ypos, DrawCmdTarget Property *611*
YRotationCenter, DrawCmdTarget Property *611*

30. EVENTCMDTARGET 612

GetXMLSettings, EventCmdTarget Function *612*
CommandList, EventCmdTarget Property *612*
Condition, EventCmdTarget Property *613*
Enable, EventCmdTarget Property *613*
EnableVariable, EventCmdTarget Property *614*
Name, EventCmdTarget Property *614*
Value, EventCmdTarget Property *615*
Variable, EventCmdTarget Property *615*

31. GAUGECMDTARGET 616

LoadExtSettings, GaugeCmdTarget Function *616*
SaveExtSettings, GaugeCmdTarget Function *616*
BarBackColor, GaugeCmdTarget Property *617*
BarBias, GaugeCmdTarget Property *617*
BarBorder, GaugeCmdTarget Property *618*
BarFillColor, GaugeCmdTarget Property *618*
BarVisible, GaugeCmdTarget Property *619*
CenterPos, GaugeCmdTarget Property *619*
ColorWarningZone, GaugeCmdTarget Property *620*
EnableWarningZone, GaugeCmdTarget Property *621*
EndAngle, GaugeCmdTarget Property *621*
EndWarningZone, GaugeCmdTarget Property *622*
ExtSettingsFile, GaugeCmdTarget Property *622*
FontHeightScale, GaugeCmdTarget Property *623*
FontNameScale, GaugeCmdTarget Property *623*
FormatVariable, GaugeCmdTarget Property *624*
GaugeMaxVariable, GaugeCmdTarget Property *624*

GaugeMinVariable, GaugeCmdTarget Property 625
GaugeType, GaugeCmdTarget Property 625
GaugeVariable, GaugeCmdTarget Property 626
GaugeWarningZoneEndVariable, GaugeCmdTarget Property 626
GaugeWarningZoneStartVariable, GaugeCmdTarget Property 627
GeneralGap, GaugeCmdTarget Property 627
InvertScale, GaugeCmdTarget Property 628
KnobBorder, GaugeCmdTarget Property 629
KnobColor, GaugeCmdTarget Property 629
LabelEvery, GaugeCmdTarget Property 630
MaxValue, GaugeCmdTarget Property 630
MinValue, GaugeCmdTarget Property 631
NeedleBorder, GaugeCmdTarget Property 632
NeedleBorderColor, GaugeCmdTarget Property 632
NeedleColor, GaugeCmdTarget Property 633
NeedleLength, GaugeCmdTarget Property 633
NeedleShadow, GaugeCmdTarget Property 634
NeedleShadowColor, GaugeCmdTarget Property 635
NeedleVisible, GaugeCmdTarget Property 635
NeedleWidth, GaugeCmdTarget Property 636
ScaleColor, GaugeCmdTarget Property 636
ScaleFormat, GaugeCmdTarget Property 637
ScaleMajorDiv, GaugeCmdTarget Property 638
ScaleMinorDiv, GaugeCmdTarget Property 638
ScaleRightBottom, GaugeCmdTarget Property 639
ScaleUnit, GaugeCmdTarget Property 639
ScaleVisible, GaugeCmdTarget Property 640
SliderBorder, GaugeCmdTarget Property 640
SliderColor, GaugeCmdTarget Property 641
SliderVisible, GaugeCmdTarget Property 642
StartAngle, GaugeCmdTarget Property 642
StartWarningZone, GaugeCmdTarget Property 643
Title, GaugeCmdTarget Property 643
TitleVisible, GaugeCmdTarget Property 644

32. GENERIC EVENTS 645

Click, Generic Event 645
DblClick, Generic Event 645
KeyDown, Generic Event 645
KeyPress, Generic Event 645
KeyUp, Generic Event 646
KillFocus, Generic Event 646
MouseDown, Generic Event 646
MouseMove, Generic Event 647
MouseUp, Generic Event 647
MouseWheel, Generic Events 648
OnChange, Generic Event 648
OnChangeExecutionCanceled, Generic Event 649
OnChangeExecutionToPromoter, Generic Event 649
OnExecutionPending, Generic Event 649
OnFireExecution, Generic Event 649
OnFireSynapse, Generic Event 650
OnGesture, Generic Event 650
OnPostPaint, Generic Event 650
OnPrePaint, Generic Event 651
OnPreSymbolLoading, Generic Event 651
OnSize, Generic Event 652
OnTextChanged, Generic Event 652
OnTextChanging, Generic Event 652
OnTimer, Generic Event 652
OnToolTip, Generic Event 653
SetFocus, Generic Event 653
SymbolLoading, Generic Event 653
SymbolUnloading, Generic Event 653

33. GRIDWNDCMDTARGET 655

OnColumnClick, GridWndCmdTarget Event *655*
OnDeleteItem, GridWndCmdTarget Event *655*
OnQueryEnd, GridWndCmdTarget Event *655*
OnQueryNext, GridWndCmdTarget Event *655*
OnQueryStart, GridWndCmdTarget Event *656*
OnSelChanged, GridWndCmdTarget Event *656*
OnSelChanging, GridWndCmdTarget Event *656*
OnSQLError, GridWndCmdTarget Event *656*
OnUpdatingDSN, GridWndCmdTarget Event *657*
AddColumn, GridWndCmdTarget Function *657*
CellEditable, GridWndCmdTarget Function *657*
DeleteColumn, GridWndCmdTarget Function *658*
DeleteRow, GridWndCmdTarget Function *658*
EditCopy, GridWndCmdTarget Function *659*
EnsureVisible, GridWndCmdTarget Function *659*
FocusCellEditable, GridWndCmdTarget Function *660*
GetColCount, GridWndCmdTarget Function *660*
GetRowCount, GridWndCmdTarget Function *661*
GetSelectedRange, GridWndCmdTarget Function *661*
EditCopy, GridWndCmdTarget Function *662*
EnsureVisible, GridWndCmdTarget Function *663*
FocusCellEditable, GridWndCmdTarget Function *663*
GetColCount, GridWndCmdTarget Function *664*
GetRowCount, GridWndCmdTarget Function *664*
GetSelectedRange, GridWndCmdTarget Function *665*
InsertRow, GridWndCmdTarget Function *666*
IsCellSelected, GridWndCmdTarget Function *666*
IsCellValid, GridWndCmdTarget Function *667*
IsCellVisible, GridWndCmdTarget Function *667*
LoadFromTextFile, GridWndCmdTarget Function *668*
LoadExtSettings, GridWndCmdTarget Function *668*
RecalcLayout, GridWndCmdTarget Function *669*
Refresh, GridWndCmdTarget Function *669*
SaveToTextFile, GridWndCmdTarget Function *670*
SaveExtSettings, GridWndCmdTarget Function *670*
SelectAll, GridWndCmdTarget Function *671*
SetSelectedRange, GridWndCmdTarget Function *671*
UpdateDatabase, GridWndCmdTarget Function *672*
UpdateVariables, GridWndCmdTarget Function *673*
AutoLayout, GridWndCmdTarget Property *673*
ButtonPos, GridWndCmdTarget Property *674*
ButtonSize, GridWndCmdTarget Property *674*
CellBkColor, GridWndCmdTarget Property *675*
CellFgColor, GridWndCmdTarget Property *676*
CellModified, GridWndCmdTarget Property *676*
CellText, GridWndCmdTarget Property *677*
Clickable, GridWndCmdTarget Property *677*
ColumnWidths, GridWndCmdTarget Property *678*
CopyBtnText, GridWndCmdTarget Property *678*
DeleteBtnText, GridWndCmdTarget Property *679*
DSN, GridWndCmdTarget Property *679*
ExtSettingsFile, GridWndCmdTarget Property *680*
FocusCellBkColor, GridWndCmdTarget Property *680*
FocusCellCol, GridWndCmdTarget Property *681*
FocusCellFgColor, GridWndCmdTarget Property *681*
FocusCellModified, GridWndCmdTarget Property *682*
FocusCellRow, GridWndCmdTarget Property *682*
FocusCellText, GridWndCmdTarget Property *683*
GraphicButtons, GridWndCmdTarget Property *683*
InsertBtnText, GridWndCmdTarget Property *684*
Promptpad,GridWndCmdTarget_Property *684*
Query, GridWndCmdTarget Property *685*
SaveBtnText, GridWndCmdTarget Property *685*
SelectAllBtnText, GridWndCmdTarget Property *686*
ShowCopyBtn, GridWndCmdTarget Property *686*
ShowDeleteBtn, GridWndCmdTarget Property *687*
ShowInsertBtn, GridWndCmdTarget Property *687*
ShowSaveBtn, GridWndCmdTarget Property *688*

ShowSelectAllBtn, GridWndCmdTarget Property 688
ShowUpdateBtn, GridWndCmdTarget Property 689
TextFileName, GridWndCmdTarget Property 689
TextSeparator, GridWndCmdTarget Property 690
UpdateBtnText, GridWndCmdTarget Property 690
UpdateVariable, GridWndCmdTarget Property 691
UserName, GridWndCmdTarget Property 691

34. HISLOGWNDCMDTARGET 693

OnFilter, HisLogWndCmdTarget Event 693
OnPrint, HisLogWndCmdTarget Event 693
OnRefresh, HisLogWndCmdTarget Event 693
EditCopy, HisLogWndCmdTarget Function 693
EditLayout, HisLogWndCmdTarget Function 694
LoadExtSettings, HisLogWndCmdTarget Function 694
RecalcLayout, HisLogWndCmdTarget Function 695
Refresh, HisLogWndCmdTarget Function 695
SaveExtSettings, HisLogWndCmdTarget Function 696
AutoLayout, HisLogWndCmdTarget Property 696
ButtonPos, HisLogWndCmdTarget Property 697
ButtonSize, HisLogWndCmdTarget Property 698
Clickable, HisLogWndCmdTarget Property 698
EventType, HisLogWndCmdTarget Property 699
ExtSettingsFile, HisLogWndCmdTarget Property 699
FilterBtnText, HisLogWndCmdTarget Property 700
FilterEvent, HisLogWndCmdTarget Property 701
FilterEventTypeCol, HisLogWndCmdTarget Property 701
FilterEventTypeColCondition, HisLogWndCmdTarget Property 702
FilterFromDate, HisLogWndCmdTarget Property 703
FilterToDate, HisLogWndCmdTarget Property 703
FilterUniqueID, HisLogWndCmdTarget Property 704
FilterUser, HisLogWndCmdTarget Property 704
FormatDateTime, HisLogWndCmdTarget Property 705
FormatDuration, HisLogWndCmdTarget Property 705
GraphicButtons, HisLogWndCmdTarget Property 706
NetworkBackupServerName, HisLogWndCmdTarget Property 707
IncludeMilliseconds, HisLogWndCmdTarget Property 707
MaxCount, HisLogWndCmdTarget Property 708
NetworkBackupServerName, HisLogWndCmdTarget Property 708
NetworkServerName, HisLogWndCmdTarget Property 709
PrintBtnText, HisLogWndCmdTarget Property 709
Project, HisLogWndCmdTarget Property 710
Query, HisLogWndCmdTarget Property 710
RefreshBtnText, HisLogWndCmdTarget Property 711
ReportFile, HisLogWndCmdTarget Property 712
ShowFilterBtn, HisLogWndCmdTarget Property 712
ShowPrintBtn, HisLogWndCmdTarget Property 713
ShowRefreshBtn, HisLogWndCmdTarget Property 713
SortBy, HisLogWndCmdTarget Property 714
SubItemComment, HisLogWndCmdTarget Property 715
SubItemCommentPos, HisLogWndCmdTarget Property 715
SubItemCommentWidth, HisLogWndCmdTarget Property 716
SubItemDesc, HisLogWndCmdTarget Property 716
SubItemDescPos, HisLogWndCmdTarget Property 717
SubItemDescWidth, HisLogWndCmdTarget Property 717
SubItemDuration, HisLogWndCmdTarget Property 718
SubItemDurationPos, HisLogWndCmdTarget Property 718
SubItemDurationWidth, HisLogWndCmdTarget Property 719
SubItemEventId, HisLogWndCmdTarget Property 719
SubItemEventIdPos, HisLogWndCmdTarget Property 720
SubItemEventIdWidth, HisLogWndCmdTarget Property 721
SubItemEventNum, HisLogWndCmdTarget Property 721
SubItemEventNumPos, HisLogWndCmdTarget Property 722
SubItemEventNumWidth, HisLogWndCmdTarget Property 722
SubItemText, HisLogWndCmdTarget Property 723
SubItemTextPos, HisLogWndCmdTarget Property 723
SubItemTextWidth, HisLogWndCmdTarget Property 724

SubItemTime, HisLogWndCmdTarget Property *724*
SubItemTimePos, HisLogWndCmdTarget Property *725*
SubItemTimeWidth, HisLogWndCmdTarget Property *725*
SubItemUser, HisLogWndCmdTarget Property *726*
SubItemUserPos, HisLogWndCmdTarget Property *726*
SubItemUserWidth, HisLogWndCmdTarget Property *727*

35. HOURSELECTORCMDTARGET 728

OnAddScheduler, HourSelectorCmdTarget Event *728*
OnCancel, HourSelectorCmdTarget Event *728*
OnRemoveScheduler, HourSelectorCmdTarget Event *728*
OnSave, HourSelectorCmdTarget Event *729*
OnSchedulerChanged, HourSelectorCmdTarget Event *729*
OnSwitchGridMode, HourSelectorCmdTarget Event *729*
OnSwitchHolidays, HourSelectorCmdTarget Event *730*
Cancel, HourSelectorCmdTarget Function *730*
LoadExtSettings, HourSelectorCmdTarget Function *731*
RecalcLayout, HourSelectorCmdTarget Function *731*
Save, HourSelectorCmdTarget Function *732*
SaveExtSettings, HourSelectorCmdTarget Function *732*
AddBtnText, HourSelectorCmdTarget Property *733*
Border, HourSelectorCmdTarget Property *733*
ButtonPos, HourSelectorCmdTarget Property *734*
ButtonSize, HourSelectorCmdTarget Property *734*
CancelBtnText, HourSelectorCmdTarget Property *735*
ColorSelCell, HourSelectorCmdTarget Property *735*
DaysText, HourSelectorCmdTarget Property *736*
EditMode, HourSelectorCmdTarget Property *736*
EndTimeColText, HourSelectorCmdTarget Property *737*
ErrorString, HourSelectorCmdTarget Property *738*
ExtSettingsFile, HourSelectorCmdTarget Property *738*
GraphicButtons, HourSelectorCmdTarget Property *739*
GridModeBtnText, HourSelectorCmdTarget Property *739*
HolidaysBtnText, HourSelectorCmdTarget Property *740*
MaxRow, HourSelectorCmdTarget Property *740*
NetworkBackupServerName, HourSelectorCmdTarget Property *741*
NetworkServer name,HourSelectorCmdTarget Property *742*
PromptPad, HourSelectorCmdTarget Property *742*
RemoveBtnText, HourSelectorCmdTarget Property *743*
SaveBtnText, HourSelectorCmdTarget Property *743*
Scheduler, HourSelectorCmdTarget Property *744*
ShowAddBtn, HourSelectorCmdTarget Property *744*
ShowCancelBtn, HourSelectorCmdTarget Property *745*
ShowColumValue, HourSelectorCmdTarget Property *745*
ShowColumVariable, HourSelectorCmdTarget Property *746*
ShowComboScheduler, HourSelectorCmdTarget Property *746*
ShowHolidaysBtn, HourSelectorCmdTarget Property *747*
ShowRemoveBtn, HourSelectorCmdTarget Property *747*
ShowSaveBtn, HourSelectorCmdTarget Property *748*
StartTimeColText, HourSelectorCmdTarget Property *749*
ValueColText, HourSelectorCmdTarget Property *749*
ValueErrorString, HourSelectorCmdTarget Property *750*
VariableColText, HourSelectorCmdTarget Property *750*

36. IOPORTINTERFACE 752

IOClosePort, IOPortInterface Function *752*
IOGetLastError, IOPortInterface Function *752*
IOInBufferCount, IOPortInterface Function *753*
IOInput, IOPortInterface Function *754*
IOOutput, IOPortInterface Function *754*
IOPortOpen, IOPortInterface Function *755*
IOBreak, IOPortInterface Property *756*
IOCDHolding, IOPortInterface Property *756*
IIOCTSHolding, IOPortInterface Property *757*
IODSRHolding, IOPortInterface Property *758*
IODTREnable, IOPortInterface Property *758*

IOInputLen, IOPortInterface Property 759
IOOutBufferCount, IOPortInterface Property 759
IORTSEnable, IOPortInterface Property 760

37. LISTBOXCMDTARGET 761

OnSelected, ListBoxCmdTarget Event 761
OnSelecting, ListBoxCmdTarget Event 761
AddString, ListBoxCmdTarget Function 761
GetCount, ListBoxCmdTarget Function 762
GetSelectedIndex, ListBoxCmdTarget Function 762
GetText, ListBoxCmdTarget Function 762
LoadExtSettings, ListBoxCmdTarget Function 763
RefillList, ListBoxCmdTarget Function 763
RemoveString, ListBoxCmdTarget Function 764
SaveExtSettings, ListBoxCmdTarget Function 764
ExtSettingsFile, ListBoxCmdTarget Property 765
ListData, ListBoxCmdTarget Property 765
ListVariable, ListBoxCmdTarget Property 766
SortItems,ListBoxCmdTarget Property 766
Variable, ListBoxCmdTarget Property 767

38. NETWORKCLIENTCMD 768

ConnectVariable, NetworkClientCmd Function 768
DisconnectVariable, NetworkClientCmd Function 769
GetClientRules, NetworkClientCmd Function 769
GetRASStation, NetworkClientCmd Function 770
IsServerAvailable, NetworkClientCmd Function 770

39. NETWORKREDUDANCYCMD 772

ActNumRetries, NetworkRedudancyCmd Function 772
CallBackServer, NetworkRedudancyCmd Function 772
ConnectToServer, NetworkRedudancyCmd Function 773
GetTotalPendingMessage, NetworkRedudancyCmd Function 773
IsActiveServer, NetworkRedudancyCmd Function 774
LastInteractionTime, NetworkRedudancyCmd Function 774
PendingStartedDriverOnSecondary, NetworkRedudancyCmd Function 775
SecondaryServerConnected, NetworkRedudancyCmd Function 775
StartedDriverOnSecondary, NetworkRedudancyCmd Function 776
StatusVariable, NetworkRedudancyCmd Function 776
Synchronizing, NetworkRedudancyCmd Function 777
Type, NetworkRedudancyCmd Function 777
DriverErrorTimeout, NetworkRedudancyCmd Property 778
MaxHisCacheHits, NetworkRedudancyCmd Property 778
Retries, NetworkRedudancyCmd Property 779
SwitchServerOnDriverError, NetworkRedudancyCmd Property 779
SyncTimeFreq, NetworkRedudancyCmd Property 780
TimeOut, NetworkRedudancyCmd Property 780

40. OPCAECMDTARGET 781

Refresh, OPCAECmdTarget Function 781
Reconnect, OPCAECmdTarget Function 781
GetXMLSettings, OPCAECmdTarget Function 781
GetServerVendorInfo, OPCAECmdTarget Function 782
GetServerStatus, OPCAECmdTarget Function 782
GetNumObjectsInHeap, OPCAECmdTarget Function 782
GetNumEventInQueue, OPCAECmdTarget Function 783
GetEventInQueueAt, OPCAECmdTarget Function 783
Server, OPCAECmdTarget Property 783
ReconnectTime, OPCAECmdTarget Property 784
Node, OPCAECmdTarget Property 784
MaxEventQueue, OPCAECmdTarget Property 784

41. OPCCLIENTCMDTARGET 785

ClearDynOPCItemTypeCache, OPCClientCmdTarget Function 785

GetOPCAEClientObject, OPCClientCmdTarget Function *785*
GetOPCDAClientObject, OPCClientCmdTarget Function *786*
GetXMLSettings, OPCClientCmdTarget Function *787*
ReconnectAll, OPCClientCmdTarget Function *787*
PoolOPCClientTest, OPCClientCmdTarget Property *788*
RefreshRateDynamicOPC, OPCClientCmdTarget Property *788*
StartupTimeout, OPCClientCmdTarget Property *789*
TimeoutDynamicOperation, OPCClientCmdTarget Property *789*
TimeoutOPCClientTest, OPCClientCmdTarget Property *790*

42. OPCCLIENTGROUPOBJCMDTARGET 791

UpdateGroupProperties, OPCClientGroupObjCmdTarget Function *791*
IsGroupConnected, OPCClientGroupObjCmdTarget Function *791*
GetXMLSettings, OPCClientGroupObjCmdTarget Function *792*
GetServerObject, OPCClientGroupObjCmdTarget Function *793*
GetNumObjectsInHeap, OPCClientGroupObjCmdTarget Function *793*
GetName, OPCClientGroupObjCmdTarget Function *794*
GetItemObject, OPCClientGroupObjCmdTarget Function *795*
ConnectGroup, OPCClientGroupObjCmdTarget Function *795*
UpdateRate, OPCClientGroupObjCmdTarget Property *796*
TimeBias, OPCClientGroupObjCmdTarget Property *797*
NotInUseRefreshRate, OPCClientGroupObjCmdTarget Property *797*
NotInUseDisableGroup, OPCClientGroupObjCmdTarget Property *798*
LocalID, OPCClientGroupObjCmdTarget Property *799*
EnableGroup, OPCClientGroupObjCmdTarget Property *799*
DeadBand, OPCClientGroupObjCmdTarget Property *800*
Active, OPCClientGroupObjCmdTarget Property *801*

43. OPCCLIENTITEMOBJCMDTARGET 802

ForceReadData, OPCClientItemObjCmdTarget Function *802*
ForceWriteData, OPCClientItemObjCmdTarget Function *802*
GetGroupObject, OPCClientItemObjCmdTarget Function *803*
GetItemQuality, OPCClientItemObjCmdTarget Function *804*
GetItemTimeStamp, OPCClientItemObjCmdTarget Function *805*
GetLinkedVariableObject, OPCClientItemObjCmdTarget Function *806*
GetNumObjectsInHeap, OPCClientItemObjCmdTarget Function *806*
GetXMLSettings, OPCClientItemObjCmdTarget Function *807*
IsItemConnected, OPCClientItemObjCmdTarget Function *808*
ReconnectItem, OPCClientItemObjCmdTarget Function *808*
EnableRead, OPCClientItemObjCmdTarget Property *809*
EnableWrite, OPCClientItemObjCmdTarget Property *810*
ItemID, OPCClientItemObjCmdTarget Property *811*
ItemPath, OPCClientItemObjCmdTarget Property *811*
LinkedVariable, OPCClientItemObjCmdTarget Property *812*
ReRead, OPCClientItemObjCmdTarget Property *813*
SyncDataAtStartup, OPCClientItemObjCmdTarget Property *813*
vtType, OPCClientItemObjCmdTarget Property *814*
WriteSync, OPCClientItemObjCmdTarget Property *815*

44. OPCCLIENTOBJCMDTARGET 817

IsConnected, OPCClientObjCmdTarget Function *817*
GetXMLSettings, OPCClientObjCmdTarget Function *817*
GetServerVendorInfo, OPCClientObjCmdTarget Function *818*
GetServerStatus, OPCClientObjCmdTarget Function *818*
GetServerName, OPCClientObjCmdTarget Function *819*
GetServerCLSID, OPCClientObjCmdTarget Function *820*
GetOPCClientDocObj, OPCClientObjCmdTarget Function *820*
GetNumObjectsInHeap, OPCClientObjCmdTarget Function *821*
GetNodeName, OPCClientObjCmdTarget Function *821*
GetGroupObject, OPCClientObjCmdTarget Function *822*
ConnectServer, OPCClientObjCmdTarget Function *823*
ReconnectTime, OPCClientObjCmdTarget Property *823*
ReReadDynamicItems, OPCClientObjCmdTarget Property *824*

45. OPCSERVERCMDTARGET 825

FireAEEvent, OPCServerCmdTarget Function *825*
GetNumServingTags, OPCServerCmdTarget Function *825*
GetServerName,OPCServerCmdTarget Function *826*
GetNumConnectedClients, OPCServerCmdTarget Function *826*
ServerStatus, OPCServerCmdTarget Property *827*

46. PMEDOCCMDTARGET 828

AckAllAlarms, PmeDocCmdTarget Function *828*
AddSysLogMessage, PmeDocCmdTarget Function *828*
CreateObjectLic, PmeDocCmdTarget Function *829*
CreateRemoteObject, PmeDocCmdTarget Function *829*
GetAlarm, PmeDocCmdTarget Function *830*
GetAlarmsPath, PmeDocCmdTarget Function *830*
GetChildProject, PmeDocCmdTarget Function *831*
GetCurrentListAlarms, PmeDocCmdTarget Function *831*
GetDataLoggerRecipe, PmeDocCmdTarget Function *832*
GetDataLoggerRecipePath, PmeDocCmdTarget Function *833*
GetDataPath, PmeDocCmdTarget Function *833*
GetDrawingPath, PmeDocCmdTarget Function *833*
GetDriverInterface, PmeDocCmdTarget Function *834*
GetEvent, PmeDocCmdTarget Function *834*
GetFatherProject, PmeDocCmdTarget Function *835*
GetHisLogADOConn, PmeDocCmdTarget Function *835*
GetHisLogDNSConnectionString, PmeDocCmdTarget Function *836*
GetIOPortInterface, PmeDocCmdTarget Function *837*
GetLastAlarmOn, PmeDocCmdTarget Function *837*
GetLogPath, PmeDocCmdTarget Function *838*
GetNetworkClient, PmeDocCmdTarget Function *838*
GetNetworkRedundancy, PmeDocCmdTarget Function *839*
GetNetworkServer, PmeDocCmdTarget Function *839*
GetNetworkUserLogPath, PmeDocCmdTarget Function *840*
GetOPCClient, PmeDocCmdTarget Function *840*
GetOPCServer, PmeDocCmdTarget Function *841*
GetProjectFileName, PmeDocCmdTarget Function *841*
GetProjectTitle, PmeDocCmdTarget Function *842*
GetRealTimeDB, PmeDocCmdTarget Function *842*
GetResourcePath, PmeDocCmdTarget Function *842*
GetScaling, PmeDocCmdTarget Function *843*
GetScheduler, PmeDocCmdTarget Function *843*
GetSynopticInterface, PmeDocCmdTarget Function *844*
GetUserAndGroup, PmeDocCmdTarget Function *845*
IsAlarmAreaActive, PmeDocCmdTarget Function *845*
IsAlarmAreaON, PmeDocCmdTarget Function *846*
IsChildProject, PmeDocCmdTarget Function *846*
IsInStoppingMode, PmeDocCmdTarget Function *847*
IsRunning, PmeDocCmdTarget Function *847*
PlaySoundFile, PmeDocCmdTarget Function *848*
ResetAllAlarms, PmeDocCmdTarget Function *848*
RunningOnCE, PmeDocCmdTarget Function *849*
RunScript, PmeDocCmdTarget Function *849*
SendDispatcherMessage, PmeDocCmdTarget Function *850*
StartAlarmDispatcher, PmeDocCmdTarget Function *850*
StopPlaySoundFile, PmeDocCmdTarget Function *851*
UnloadScript, PmeDocCmdTarget Function *851*
ActiveLanguage, PmeDocCmdTarget Property *852*
ChildProjectActiveNetworkServer, PmeDocCmdTarget Property *852*
ChildProjectBackupNetworkServer, PmeDocCmdTarget Property *853*
ChildProjectName, PmeDocCmdTarget Property *853*
ChildProjectNetworkServer, PmeDocCmdTarget Property *854*
ChildProjectStartable, PmeDocCmdTarget Property *854*
HisLogAlarmDurationDays, PmeDocCmdTarget Property *855*
HisLogAlarmDurationHours, PmeDocCmdTarget Property *855*
HisLogAlarmDurationMinutes, PmeDocCmdTarget Property *856*
HisLogAlarmTable, PmeDocCmdTarget Property *856*
HisLogCommentColName, PmeDocCmdTarget Property *857*
HisLogDefVarCharPrecision, PmeDocCmdTarget Property *857*
HisLogDescriptionColName, PmeDocCmdTarget Property *858*

HisLogDriverDurationDays, PmeDocCmdTarget Property *858*
HisLogDriverDurationHours, PmeDocCmdTarget Property *859*
HisLogDriverDurationMinutes, PmeDocCmdTarget Property *859*
HisLogDriverTable, PmeDocCmdTarget Property *860*
HisLogDsn, PmeDocCmdTarget Property *860*
HisLogDurationColName, PmeDocCmdTarget Property *861*
HisLogEventTypeColName, PmeDocCmdTarget Property *861*
HisLogEventTypeNumColName, PmeDocCmdTarget Property *862*
HisLogLocalTimeColName, PmeDocCmdTarget Property *863*
HisLogMaxCacheBeforeFlush, PmeDocCmdTarget Property *863*
HisLogMaxError, PmeDocCmdTarget Property *864*
HisLogMaxNumberTrans, PmeDocCmdTarget Property *864*
HisLogMSecColName, PmeDocCmdTarget Property *865*
HisLogRecycleDBConnection, PmeDocCmdTarget Property *865*
HisLogSubEventTypeColName, PmeDocCmdTarget Property *866*
HisLogSysTable, PmeDocCmdTarget Property *866*
HisLogSystemsDurationDays, PmeDocCmdTarget Property *867*
HisLogSystemsDurationHours, PmeDocCmdTarget Property *867*
HisLogSystemsDurationMinutes, PmeDocCmdTarget Property *868*
HisLogTimeColName, PmeDocCmdTarget Property *868*
HisLogTransactionIDColName, PmeDocCmdTarget Property *869*
HisLogUniqueIDColName, PmeDocCmdTarget Property *869*
HisLogUseIMDB, PmeDocCmdTarget Property *870*
HisLogUser, PmeDocCmdTarget Property *870*
HisLogUserColName, PmeDocCmdTarget Property *871*
ShutdownScript, PmeDocCmdTarget Property *871*
StartChildProjectWithFather, PmeDocCmdTarget Property *872*
StartupScreen, PmeDocCmdTarget Property *872*
StartupScript, PmeDocCmdTarget Property *873*
StoreCryptProject, PmeDocCmdTarget Property *873*
StoreCryptProjectResources, PmeDocCmdTarget Property *874*
StoreCryptProjectStrings, PmeDocCmdTarget Property *874*
StoreUnicodeProject, PmeDocCmdTarget Property *875*
StoreZippedProject, PmeDocCmdTarget Property *875*
StringFromID, PmeDocCmdTarget Property *876*
TargetClientJ2ME, PmeDocCmdTarget Property *876*
TargetClientJ2SE, PmeDocCmdTarget Property *877*
TargetClientWin32, PmeDocCmdTarget Property *877*
TargetClientWinCE, PmeDocCmdTarget Property *878*
TargetPlatformWin32, PmeDocCmdTarget Property *878*
TargetPlatformWinCE, PmeDocCmdTarget Property *879*

47. RASSTATIONINTERFACE 880

GetXMLSettings, RASStationInterface Function *880*
ConnectionVariable, RASStationInterface Property *880*
DisconnectAfterSecs, RASStationInterface Property *881*
EndConnectionTime, RASStationInterface Property *881*
IsConnected, RASStationInterface Property *882*
LastConnectionTime, RASStationInterface Property *883*
LastRASErrorNumber, RASStationInterface Property *883*
LastRASErrorString, RASStationInterface Property *884*
NumRetries, RASStationInterface Property *884*
Password, RASStationInterface Property *885*
PhoneBookEntry, RASStationInterface Property *886*
PhoneNumber, RASStationInterface Property *886*
PromptForConnection, RASStationInterface Property *887*
RetryAfterSecs, RASStationInterface Property *887*
ShowConnectionDlg, RASStationInterface Property *888*
StartConnectionTime, RASStationInterface Property *889*
TotalConnectionTime, RASStationInterface Property *889*
UserName, RASStationInterface Property *890*

48. RECIPEWNDCMDTARGET 891

OnActivateRecipe, RecipeWndCmdTarget Event *891*
OnCopyRecipe, RecipeWndCmdTarget Event *891*
OnDeleteRecipe, RecipeWndCmdTarget Event *891*

OnExportRecipe, RecipeWndCmdTarget Event *891*
OnImportRecipe, RecipeWndCmdTarget Event *892*
OnPasteRecipe, RecipeWndCmdTarget Event *892*
OnPrintRecipe, RecipeWndCmdTarget Event *892*
OnReadRecipe, RecipeWndCmdTarget Event *892*
OnRecipeIndexChanged, RecipeWndCmdTarget Event *892*
OnRefreshRecipe, RecipeWndCmdTarget Event *893*
OnSaveRecipe, RecipeWndCmdTarget Event *893*
EditLayout, RecipeWndCmdTarget Function *893*
ExportRecipeToCSV, RecipeWndCmdTarget Function *894*
ImportRecipeFromCSV, RecipeWndCmdTarget Function *894*
LoadExtSettings, RecipeWndCmdTarget Function *895*
RecalcLayout, RecipeWndCmdTarget Function *896*
Reconnect, RecipeWndCmdTarget Function *896*
Refresh, RecipeWndCmdTarget Function *897*
SaveExtSettings, RecipeWndCmdTarget Function *897*
ActivateBtnText, RecipeWndCmdTarget Property *898*
ActivateMessage, RecipeWndCmdTarget Property *898*
AutoLayout, RecipeWndCmdTarget Property *899*
ButtonPos, RecipeWndCmdTarget Property *899*
ButtonSize, RecipeWndCmdTarget Property *900*
Clickable, RecipeWndCmdTarget Property *900*
CopyBtnText, RecipeWndCmdTarget Property *901*
CurrentRecipeVariable, RecipeWndCmdTarget Property *901*
DeleteBtnText, RecipeWndCmdTarget Property *902*
DeleteMessage, RecipeWndCmdTarget Property *902*
ErrorString, RecipeWndCmdTarget Property *903*
ExportBtnText, RecipeWndCmdTarget Property *903*
ExtSettingsFile, RecipeWndCmdTarget Property *904*
GraphicButtons, RecipeWndCmdTarget Property *904*
ImpExpSeparator, RecipeWndCmdTarget Property *905*
ImportBtnText, RecipeWndCmdTarget Property *905*
NetworkBackupServerName, RecipeWndCmdTarget Property *906*
NetworkServerName, RecipeWndCmdTarget Property *907*
PasteBtnText, RecipeWndCmdTarget Property *907*
Project, RecipeWndCmdTarget Property *908*
PromptPad, RecipeWndCmdTarget Property *908*
PrintBtnText, RecipeWndCmdTarget Property *909*
ReadBtnText, RecipeWndCmdTarget Property *909*
Recipe, RecipeWndCmdTarget Property *910*
RefreshBtnText, RecipeWndCmdTarget Property *910*
SaveBtnText, RecipeWndCmdTarget Property *911*
SaveMessage, RecipeWndCmdTarget Property *911*
ShowActivateBtn, RecipeWndCmdTarget Property *912*
ShowCopyBtn, RecipeWndCmdTarget Property *912*
ShowDeleteBtn, RecipeWndCmdTarget Property *913*
ShowExportBtn, RecipeWndCmdTarget Property *913*
ShowImportBtn, RecipeWndCmdTarget Property *914*
ShowPasteBtn, RecipeWndCmdTarget Property *914*
ShowPrintBtn, RecipeWndCmdTarget Property *915*
ShowReadBtn, RecipeWndCmdTarget Property *915*
ShowRefreshBtn, RecipeWndCmdTarget Property *916*
ShowSaveBtn, RecipeWndCmdTarget Property *916*
SubItemDescription, RecipeWndCmdTarget Property *917*
SubItemDescriptionPos, RecipeWndCmdTarget Property *917*
SubItemDescriptionWidth, RecipeWndCmdTarget Property *918*
SubItemMax, RecipeWndCmdTarget Property *918*
SubItemMaxPos, RecipeWndCmdTarget Property *919*
SubItemMaxWidth, RecipeWndCmdTarget Property *919*
SubItemMin, RecipeWndCmdTarget Property *920*
SubItemMinPos, RecipeWndCmdTarget Property *920*
SubItemMinWidth, RecipeWndCmdTarget Property *921*
SubItemUnits, RecipeWndCmdTarget Property *921*
SubItemUnitsPos, RecipeWndCmdTarget Property *922*
SubItemUnitsWidth, RecipeWndCmdTarget Property *922*
SubItemValue, RecipeWndCmdTarget Property *923*
SubItemValuePos, RecipeWndCmdTarget Property *923*
SubItemValueWidth, RecipeWndCmdTarget Property *924*

SubItemVariable, RecipeWndCmdTarget Property *924*
SubItemVariablePos, RecipeWndCmdTarget Property *925*
SubItemVariableWidth, RecipeWndCmdTarget Property *925*

49. SCALINGCMDTARGET 927

GetXMLSettings, ScalingCmdTarget Function *927*
Reinit, ScalingCmdTarget Function *927*
DeadBandValue, ScalingCmdTarget Property *928*
Enabled, ScalingCmdTarget Property *928*
Name, ScalingCmdTarget Property *929*
RawMaxValue, ScalingCmdTarget Property *929*
RawMinValue, ScalingCmdTarget Property *930*
RawVariableName, ScalingCmdTarget Property *930*
ScaledMaxValue, ScalingCmdTarget Property *931*
ScaledMinValue, ScalingCmdTarget Property *931*
ScaleVariableName, ScalingCmdTarget Property *932*

50. SCHEDULERCMDTARGET 933

AddHoliday, SchedulerCmdTarget Function *933*
GetHolidaysString, SchedulerCmdTarget Function *933*
GetXMLSettings, SchedulerCmdTarget Function *934*
IsHoliday, SchedulerCmdTarget Function *934*
RemoveHoliday, SchedulerCmdTarget Function *935*
Reset, SchedulerCmdTarget Function *935*
SaveRetentive, SchedulerCmdTarget Function *936*
CommandList, SchedulerCmdTarget Property *936*
CommandListOff, SchedulerCmdTarget Property *937*
Enabled, SchedulerCmdTarget Property *937*
EnableVariable, SchedulerCmdTarget Property *938*
HasHolidays, SchedulerCmdTarget Property *938*
HolidaysPlan, SchedulerCmdTarget Property *939*
Name, SchedulerCmdTarget Property *940*
Plan, SchedulerCmdTarget Property *940*
TimeAndDate, SchedulerCmdTarget Property *941*
Type, SchedulerCmdTarget Property *942*
TreatHolidaysAsSunday, SchedulerCmdTarget Property *942*

51. SCRIPTMEINTERFACE 944

Loading, ScriptMEInterface Event *944*
Main, ScriptMEInterface Event *944*
Unloading, ScriptMEInterface Event *944*
EnterGlobalCriticalSection, ScriptMEInterface Function *944*
GetCurrentUser, ScriptMEInterface Function *945*
GetInstanceNumber, ScriptMEInterface Function *945*
GetParameter, ScriptMEInterface Function *946*
HasPreviousInstance, ScriptMEInterface Function *946*
IsStopping, ScriptMEInterface Function *947*
LeaveGlobalCriticalSection, ScriptMEInterface Function *947*
RunningOnServerSide, ScriptMEInterface Function *948*
ShowDebuggerWnd, ScriptMEInterface Function *948*
MaxInstances, ScriptMEInterface Property *949*
ModalDialog, ScriptMEInterface Property *949*
RunAtServer, ScriptMEInterface Property *950*
SeparateThread, ScriptMEInterface Property *950*
SleepExecution, ScriptMEInterface Property *950*
StatusVariable, ScriptMEInterface Property *951*
SyncroScriptTimeout, ScriptMEInterface Property *951*
ThreadPriority, ScriptMEInterface Property *952*
UseOwnTrace, ScriptMEInterface Property *952*
UseUIInterface, ScriptMEInterface Property *953*

52. SYNOPTICCMDTARGET 954

Click, Generic Event *954*
DblClick, Generic Event *954*
KeyDown, Generic Event *954*

KeyPress, Generic Event *954*
KeyUp, Generic Event *955*
KillFocus, Generic Event *955*
MouseDown, Generic Event *955*
MouseMove, Generic Event *956*
MouseUp, Generic Event *956*
OnActivate, SynopticCmdTarget Event *957*
OnQueryEndSession, SynopticCmdTarget Event *957*
OnSize, SynopticCmdTarget Event *957*
OnStartSynapsisExecution, SynopticCmdTarget Event *957*
OnStopSynapsisExecution, SynopticCmdTarget Event *958*
SetFocus, Generic Event *958*
SynopticLoading, SynopticCmdTarget Event *958*
SynopticUnloading, SynopticCmdTarget Event *958*
CloseSynoptic, SynopticCmdTarget Function *959*
CreateNewSymbol, SynopticCmdTarget Function *959*
DestroySymbol, SynopticCmdTarget Function *960*
GetAbsoluteSubObject, SynopticCmdTarget Function *961*
GetActiveUserObject, SynopticCmdTarget Function *962*
GetAlias, SynopticCmdTarget Function *962*
GetAliasListName, SynopticCmdTarget Function *963*
GetAliasListValue, SynopticCmdTarget Function *963*
GetAppTimeZone, PmeDocCmdTarget Function *964*
GetFocus, SynopticCmdTarget Function *964*
GetImage, SynopticCmdTarget Function *965*
GetInstanceNumber, SynopticCmdTarget Function *966*
GetObjectByUniqueID, SynopticCmdTarget Function *966*
GetParameterVariable, SynopticCmdTarget Function *967*
GetSubObject, SynopticCmdTarget Function *967*
GetTimeZone, SynopticCmdTarget Function *968*
GetWindowPos, SynopticCmdTarget Function *968*
IsRemoteClientView, SynopticCmdTarget Function *970*
PrintSynoptic, SynopticCmdTarget Function *970*
RemoveAlias, SynopticCmdTarget Function *971*
RemoveAllAliases, SynopticCmdTarget Function *971*
SaveImageToFile, SynopticCmdTarget Function *972*
SetAlias, SynopticCmdTarget Function *973*
SetFocusTo, SynopticCmdTarget Function *973*
SetRedraw, SynopticCmdTarget Function *974*
SetSynapsisVisible, SynopticCmdTarget Function *974*
SetWindowPos, SynopticCmdTarget Function *975*
ZoomIn, SynopticCmdTarget Function *975*
ZoomOut, SynopticCmdTarget Function *976*
ZoomTo, SynopticCmdTarget Function *976*
BackColor, SynopticCmdTarget Property *976*
BackGroundFileBitmap, SynopticCmdTarget Property *977*
BackGroundFileBitmapTile, SynopticCmdTarget Property *978*
CXBackImage, SynopticCmdTarget Property *979*
CYBackImage, SynopticCmdTarget Property *979*
CyclicExecution, SynopticCmdTarget Property *980*
EnableScrollBars, SynopticCmdTarget Property *981*
FastTickCounter, SynopticCmdTarget Property *981*
FastTickFrequency, SynopticCmdTarget Property *982*
FitInWindow, SynopticCmdTarget Property *982*
FrameTitle, SynopticCmdTarget Property *983*
GlobalContainerName, SynopticCmdTarget Property *983*
GradientColor, SynopticCmdTarget Property *984*
GradientFill, SynopticCmdTarget Property *984*
LayerVariable, SynopticCmdTarget Property *985*
MaxInstances, SynopticCmdTarget Property *986*
NonDestroyable, SynopticCmdTarget Property *986*
NumColors, SynopticCmdTarget Property *987*
ParameterFileName, SynopticCmdTarget Property *987*
ScrollPositionX, SynopticCmdTarget Property *988*
ScrollPositionY, SynopticCmdTarget Property *988*
SeparateThread, SynopticCmdTarget Property *989*
ShowOnMDITabsFlag, SynopticCmdTarget Property *989*
SlowTickCounter, SynopticCmdTarget Property *990*

SlowTickFrequency, SynopticCmdTarget Property *991*
SynapsisExecution, SynopticCmdTarget Property *991*
SynopticHeight, SynopticCmdTarget Property *992*
SynopticID, SynopticCmdTarget Property *992*
SynopticPublicSource, SynopticCmdTarget Property *993*
SynopticWidth, SynopticCmdTarget Property *993*
UseAntialaising, SynopticCmdTarget Property *994*
XBackImage, SynopticCmdTarget Property *994*
YBackImage, SynopticCmdTarget Property *995*
ZoomFactorX, SynopticCmdTarget Property *995*
ZoomFactorY, SynopticCmdTarget Property *996*

53. TRACEDBWNDCMDTARGET 997

OnFilter, TraceDBWndCmdTarget Event *997*
OnPrint, TraceDBWndCmdTarget Event *997*
OnRefresh, TraceDBWndCmdTarget Event *997*
EditCopy, TraceDBWndCmdTarget Function *997*
EditLayout, TraceDBWndCmdTarget Function *998*
LoadExtSettings, TraceDBWndCmdTarget Function *998*
RecalcLayout, TraceDBWndCmdTarget Function *999*
Refresh, TraceDBWndCmdTarget Function *999*
SaveExtSettings, TraceDBWndCmdTarget Function *1000*
AutoLayout, TraceDBWndCmdTarget Property *1000*
ButtonPos, TraceDBWndCmdTarget Property *1001*
ButtonSize, TraceDBWndCmdTarget Property *1002*
Clickable, TraceDBWndCmdTarget Property *1002*
ExtSettingsFile, TraceDBWndCmdTarget Property *1003*
FileReport, TraceDBWndCmdTarget Property *1003*
FilterBtnText, TraceDBWndCmdTarget Property *1004*
FilterFromDate, TraceDBWndCmdTarget Property *1004*
FilterToDate, TraceDBWndCmdTarget Property *1005*
FilterUser, TraceDBWndCmdTarget Property *1006*
GraphicButtons, TraceDBWndCmdTarget Property *1006*
IncludeMilliseconds, TraceDBWndCmdTarget Property *1007*
MaxCount, TraceDBWndCmdTarget Property *1007*
NetworkBackupServerName, TraceDBWndCmdTarget Property *1008*
NetworkServerName, TraceDBWndCmdTarget Property *1008*
PrintBtnText, TraceDBWndCmdTarget Property *1009*
Project, TraceDBWndCmdTarget Property *1009*
Query, TraceDBWndCmdTarget Property *1010*
RefreshBtnText, TraceDBWndCmdTarget Property *1011*
ShowFilterBtn, TraceDBWndCmdTarget Property *1011*
ShowPrintBtn, TraceDBWndCmdTarget Property *1012*
ShowRefreshBtn, TraceDBWndCmdTarget Property *1012*
SortBy, TraceDBWndCmdTarget Property *1013*
SubItemAction, TraceDBWndCmdTarget Property *1014*
SubItemActionPos, TraceDBWndCmdTarget Property *1014*
SubItemActionWidth, TraceDBWndCmdTarget Property *1015*
SubItemAfter, TraceDBWndCmdTarget Property *1015*
SubItemAfterPos, TraceDBWndCmdTarget Property *1016*
SubItemAfterWidth, TraceDBWndCmdTarget Property *1016*
SubItemBefore, TraceDBWndCmdTarget Property *1017*
SubItemBeforePos, TraceDBWndCmdTarget Property *1017*
SubItemBeforeWidth, TraceDBWndCmdTarget Property *1018*
SubItemQuality, TraceDBWndCmdTarget Property *1018*
SubItemQualityPos, TraceDBWndCmdTarget Property *1019*
SubItemQualityWidth, TraceDBWndCmdTarget Property *1019*
SubItemTime, TraceDBWndCmdTarget Property *1020*
SubItemTimePos, TraceDBWndCmdTarget Property *1020*
SubItemTimeStamp, TraceDBWndCmdTarget Property *1021*
SubItemTimeStampPos, TraceDBWndCmdTarget Property *1022*
SubItemTimeStampWidth, TraceDBWndCmdTarget Property *1022*
SubItemTimeWidth, TraceDBWndCmdTarget Property *1023*
SubItemUser, TraceDBWndCmdTarget Property *1023*
SubItemUserPos, TraceDBWndCmdTarget Property *1024*
SubItemUserWidth, TraceDBWndCmdTarget Property *1024*
SubItemValue, TraceDBWndCmdTarget Property *1025*

SubItemValuePos, TraceDBWndCmdTarget Property *1025*
SubItemValueWidth, TraceDBWndCmdTarget Property *1026*
SubItemVarDesc, TraceDBWndCmdTarget Property *1026*
SubItemVarDescPos, TraceDBWndCmdTarget Property *1027*
SubItemVarDescWidth, TraceDBWndCmdTarget Property *1027*
SubItemVarGroup, TraceDBWndCmdTarget Property *1028*
SubItemVarGroupPos, TraceDBWndCmdTarget Property *1029*
SubItemVarGroupWidth, TraceDBWndCmdTarget Property *1029*
SubItemVarName, TraceDBWndCmdTarget Property *1030*
SubItemVarNamePos, TraceDBWndCmdTarget Property *1030*
SubItemVarNameWidth, TraceDBWndCmdTarget Property *1031*
Variable, TraceDBWndCmdTarget Property *1031*

54. TRENDCMDTARGET 1033

OnChangingState, TrendCmdTarget Event *1033*
OnCursorPosChanged, TrendCmdTarget Event *1033*
OnErrorRecordset, TrendCmdTarget Event *1033*
OnExpand, TrendCmdTarget Event *1033*
OnExportEnd, TrendCmdTarget Event *1034*
OnExportNext, TrendCmdTarget Event *1034*
OnExportStart, TrendCmdTarget Event *1034*
OnFailedCreatingThread, TrendCmdTarget Event *1035*
OnImportEnd, TrendCmdTarget Event *1035*
OnImportNext, TrendCmdTarget Event *1035*
OnImportStart, TrendCmdTarget Event *1035*
OnNext, TrendCmdTarget Event *1036*
OnPageChanged, TrendCmdTarget Event *1036*
OnPageEnd, TrendCmdTarget Event *1036*
OnPageNext, TrendCmdTarget Event *1037*
OnPagePrev, TrendCmdTarget Event *1037*
OnPageStart, TrendCmdTarget Event *1037*
OnPositionScrolled, TrendCmdTarget Event *1038*
OnPrev, TrendCmdTarget Event *1038*
OnPrint, TrendCmdTarget Event *1038*
OnRecordsetMoveNext, TrendCmdTarget Event *1038*
OnRecordsetQueryEnd, TrendCmdTarget Event *1039*
OnRecordsetQueryStart, TrendCmdTarget Event *1039*
OnResetZoom, TrendCmdTarget Event *1039*
OnStartRecording, TrendCmdTarget Event *1040*
OnStartZoom, TrendCmdTarget Event *1040*
OnStopRecording, TrendCmdTarget Event *1040*
OnUpdateData, TrendCmdTarget Event *1041*
OnZoomAreaChanged, TrendCmdTarget Event *1041*
AddPen, TrendCmdTarget Function *1041*
ClearAllSavedValues, TrendCmdTarget Function *1042*
ClearSavedValues, TrendCmdTarget Function *1043*
CloseBackupLink, TrendCmdTarget Function *1043*
CopyLegendToClipboard, TrendCmdTarget Function *1044*
EditPenProperties, TrendCmdTarget Function *1044*
ExportToClipboard, TrendCmdTarget Function *1045*
ExportToFile, TrendCmdTarget Function *1046*
GetCurrentDataLoggerName, TrendCmdTarget Function *1046*
GetCursorDateTime, TrendCmdTarget Function *1047*
GetCursorDateTimeMsec, TrendCmdTarget Function *1048*
GetCursorDateTimeString, TrendCmdTarget Function *1048*
GetCursorPenValue, TrendCmdTarget Function *1049*
GetCursorPosInLegendArea, TrendCmdTarget Function *1049*
GetCursorPosInPenArea, TrendCmdTarget Function *1050*
GetCursorPosInScaleArea, TrendCmdTarget Function *1051*
GetCursorPosInTimeArea, TrendCmdTarget Function *1051*
GetCursorPosInTrendArea, TrendCmdTarget Function *1052*
GetDateTimeColumnName, TrendCmdTarget Function *1052*
GetFirstValidDateTime, TrendCmdTarget Function *1053*
GetFirstValidDateTimeMs, TrendCmdTarget Function *1054*
GetLastValidDateTime, TrendCmdTarget Function *1054*
GetLastValidDateTimeMs, TrendCmdTarget Function *1055*
GetLastValidValuePosition, TrendCmdTarget Function *1055*

GetMaxPage, TrendCmdTarget Function *1056*
GetPenNameFromList, TrendCmdTarget Function *1057*
GetPensNumber, TrendCmdTarget Function *1057*
GetTimeFontOrientation, TrendCmdTarget Function *1058*
GetZoomAreaDateTimeFrom, TrendCmdTarget Function *1058*
GetZoomAreaDateTimeMsecFrom, TrendCmdTarget Function *1059*
GetZoomAreaDateTimeMsecTo, TrendCmdTarget Function *1059*
GetZoomAreaDateTimeStringFrom, TrendCmdTarget Function *1060*
GetZoomAreaDateTimeStringTo, TrendCmdTarget Function *1060*
GetZoomAreaDateTimeTo, TrendCmdTarget Function *1061*
GetZoomAreaScaleValueFrom, TrendCmdTarget Function *1061*
GetZoomAreaScaleValueTo, TrendCmdTarget Function *1062*
ImportFromClipboard, TrendCmdTarget Function *1063*
ImportFromFile, TrendCmdTarget Function *1063*
IsCursorPosInLegendArea, TrendCmdTarget Function *1064*
IsCursorPosInPenArea, TrendCmdTarget Function *1065*
IsCursorPosInScaleArea, TrendCmdTarget Function *1065*
IsCursorPosInTimeArea, TrendCmdTarget Function *1066*
IsCursorPosInTrendArea, TrendCmdTarget Function *1066*
LinkToDataLogger, TrendCmdTarget Function *1067*
LoadExtSettings, TrendCmdTarget Function *1067*
MoveCursorToMousePos, TrendCmdTarget Function *1068*
PageEnd, TrendCmdTarget Function *1069*
PageNext, TrendCmdTarget Function *1069*
PagePrev, TrendCmdTarget Function *1070*
PageStart, TrendCmdTarget Function *1070*
PrintTrend, TrendCmdTarget Function *1071*
Refresh, TrendCmdTarget Function *1072*
RemovePen, TrendCmdTarget Function *1072*
Requery, TrendCmdTarget Function *1073*
ResetZoom, TrendCmdTarget Function *1073*
RestartStatistic, TrendCmdTarget Function *1074*
SaveAllCurrentValue, TrendCmdTarget Function *1074*
SaveCurrentValue, TrendCmdTarget Function *1075*
SaveExtSettings, TrendCmdTarget Function *1075*
ScrollPosNext, TrendCmdTarget Function *1076*
ScrollPosPrev, TrendCmdTarget Function *1076*
SetSamplesValue, TrendCmdTarget Function *1077*
SetTimeFontOrientation, TrendCmdTarget Function *1078*
StartPanMode, TrendCmdTarget Function *1078*
StartZoomMode, TrendCmdTarget Function *1079*
AllBtnText, TrendCmdTarget Property *1079*
BackupLink, TrendCmdTarget Property *1080*
BorderLegend, TrendCmdTarget Property *1080*
BorderLegendRaised, TrendCmdTarget Property *1081*
BorderPen, TrendCmdTarget Property *1081*
BorderPenRaised, TrendCmdTarget Property *1082*
BorderTime, TrendCmdTarget Property *1082*
BorderTimeRaised, TrendCmdTarget Property *1083*
BorderTrend, TrendCmdTarget Property *1083*
BorderTrendRaised, TrendCmdTarget Property *1084*
BrushColor, TrendCmdTarget Property *1084*
ButtonPos, TrendCmdTarget Property *1085*
ButtonSize, TrendCmdTarget Property *1086*
ColumnSeparator, TrendCmdTarget Property *1086*
CompareTimeFrameBtnColor, TrendCmdTarget Property *1087*
ComposedFileName, TrendCmdTarget Property *1087*
CompressData, TrendCmdTarget Property *1088*
CurrentSelectedPen, TrendCmdTarget Property *1088*
CurrentMultiplier, TrendCmdTarget Property *1089*
CurrentTopPen, TrendCmdTarget Property *1089*
CursorPos, TrendCmdTarget Property *1090*
DataDefaultQuery, TrendCmdTarget Property *1090*
DataFileName, TrendCmdTarget Property *1091*
DataFilterBy, TrendCmdTarget Property *1092*
DataSortBy, TrendCmdTarget Property *1092*
dateFrom, TrendCmdTarget Property *1093*
dateFromCompare, TrendCmdTarget Property *1094*

DateFromCompareCurrent, TrendCmdTarget Property *1094*
DateFromCurrent, TrendCmdTarget Property *1095*
dateTo, TrendCmdTarget Property *1095*
dateToCompare, TrendCmdTarget Property *1096*
DateToCompareCurrent, TrendCmdTarget Property *1096*
DateToCurrent, TrendCmdTarget Property *1097*
DayBtnText, TrendCmdTarget Property *1097*
DrawGridAfter, TrendCmdTarget Property *1098*
ExpandBtnText, TrendCmdTarget Property *1099*
ExtSettingsFile, TrendCmdTarget Property *1099*
FontHeightLegend, TrendCmdTarget Property *1100*
FontHeightScale, TrendCmdTarget Property *1100*
FontHeightTime, TrendCmdTarget Property *1101*
FontNameLegend, TrendCmdTarget Property *1101*
FontNameScale, TrendCmdTarget Property *1102*
FontNameTime, TrendCmdTarget Property *1102*
FormatTime, TrendCmdTarget Property *1103*
Freezed, TrendCmdTarget Property *1103*
GeneralGap, TrendCmdTarget Property *1104*
HourBtnText, TrendCmdTarget Property *1104*
HourRecTime, TrendCmdTarget Property *1105*
HourViewTime, TrendCmdTarget Property *1105*
LegendBrushColor, TrendCmdTarget Property *1106*
LegendBrushVisible, TrendCmdTarget Property *1106*
LinkedDataLogger, TrendCmdTarget Property *1107*
MaxFileLength, TrendCmdTarget Property *1107*
MaxLegendVisiblePen, TrendCmdTarget Property *1108*
MaxNumFiles, TrendCmdTarget Property *1108*
MeasureBtnText, TrendCmdTarget Property *1109*
MeasureTextColor, TrendCmdTarget Property *1109*
MinBtnText, TrendCmdTarget Property *1110*
MinRecTime, TrendCmdTarget Property *1111*
MinViewTime, TrendCmdTarget Property *1111*
MonthBtnText, TrendCmdTarget Property *1112*
MsecRecTime, TrendCmdTarget Property *1112*
NetworkBackupServerName, TrendCmdTarget Property *1113*
NetworkServerName, TrendCmdTarget Property *1113*
NextBtnText, TrendCmdTarget Property *1114*
NoneBtnText, TrendCmdTarget Property *1114*
NumCacheRecordFile, TrendCmdTarget Property *1115*
NumXGridDivision, TrendCmdTarget Property *1116*
NumXMinorGridDivision, TrendCmdTarget Property *1116*
NumYGridDivision, TrendCmdTarget Property *1117*
NumYMinorGridDivision, TrendCmdTarget Property *1117*
Page, TrendCmdTarget Property *1118*
PageNextBtnText, TrendCmdTarget Property *1118*
PagePrevBtnText, TrendCmdTarget Property *1119*
PauseRunBtnText, TrendCmdTarget Property *1119*
PenAutoscale, TrendCmdTarget Property *1120*
PenAverageLineColor, TrendCmdTarget Property *1120*
PenBackBrushPattern, TrendCmdTarget Property *1121*
PenBrushColor, TrendCmdTarget Property *1121*
PenBrushVisible, TrendCmdTarget Property *1122*
PenColor, TrendCmdTarget Property *1122*
PenDLColumnName, TrendCmdTarget Property *1123*
PenDLRName, TrendCmdTarget Property *1124*
PenEditable, TrendCmdTarget Property *1124*
PenFormatScale, TrendCmdTarget Property *1125*
PenLogarithmicScale, TrendCmdTarget Property *1125*
PenMaxLineColor, TrendCmdTarget Property *1126*
PenMaxValue, TrendCmdTarget Property *1126*
PenMinLineColor, TrendCmdTarget Property *1127*
PenMinValue, TrendCmdTarget Property *1127*
PenPlotType, TrendCmdTarget Property *1128*
PenScaleRightBottom, TrendCmdTarget Property *1128*
PenShowAverageLine, TrendCmdTarget Property *1129*
PenShowMaxLine, TrendCmdTarget Property *1129*
PenShowMinLine, TrendCmdTarget Property *1130*

PenShowScale, TrendCmdTarget Property *1130*
PenSize, TrendCmdTarget Property *1131*
PenSize, TrendCmdTarget Property *1132*
PenStyle, TrendCmdTarget Property *1132*
PenVariable, TrendCmdTarget Property *1133*
PenWidth, TrendCmdTarget Property *1133*
PrevBtnText, TrendCmdTarget Property *1134*
PrintBtnText, TrendCmdTarget Property *1134*
Recording, TrendCmdTarget Property *1135*
RecordOnFile, TrendCmdTarget Property *1136*
SampleDateTime, TrendCmdTarget Property *1136*
SampleDateTimeMs, TrendCmdTarget Property *1137*
SamplePerUpdate, TrendCmdTarget Property *1137*
Samples, TrendCmdTarget Property *1138*
SampleValue, TrendCmdTarget Property *1138*
ScrollPosition, TrendCmdTarget Property *1139*
SecBtnText, TrendCmdTarget Property *1140*
SecRecTime, TrendCmdTarget Property *1140*
SecViewTime, TrendCmdTarget Property *1141*
ShiftGrid, TrendCmdTarget Property *1141*
ShowBreakLines, TrendCmdTarget Property *1142*
ShowCompareTimeFrameBtn, TrendCmdTarget Property *1142*
ShowDate, TrendCmdTarget Property *1143*
ShowExpandBtn, TrendCmdTarget Property *1143*
ShowFirstPointBtn, TrendCmdTarget Property *1144*
ShowLastPointBtn, TrendCmdTarget Property *1144*
ShowLegend, TrendCmdTarget Property *1145*
ShowMeasureBtn, TrendCmdTarget Property *1145*
ShowMinorXGrid, TrendCmdTarget Property *1146*
ShowMinorYGrid, TrendCmdTarget Property *1146*
ShowMsec, TrendCmdTarget Property *1147*
ShowNextBtn, TrendCmdTarget Property *1147*
ShowNextPointBtn, TrendCmdTarget Property *1148*
ShowPageNextBtn, TrendCmdTarget Property *1148*
ShowPagePrevBtn, TrendCmdTarget Property *1149*
ShowPauseRunBtn, TrendCmdTarget Property *1150*
ShowPen, TrendCmdTarget Property *1150*
ShowPenLabels, TrendCmdTarget Property *1151*
ShowPenPoints, TrendCmdTarget Property *1151*
ShowPrevBtn, TrendCmdTarget Property *1152*
ShowPrevPointBtn, TrendCmdTarget Property *1152*
ShowPrintBtn, TrendCmdTarget Property *1153*
ShowSavedValues, TrendCmdTarget Property *1154*
ShowTime, TrendCmdTarget Property *1154*
ShowTimeFrameBtn, TrendCmdTarget Property *1155*
ShowTitle, TrendCmdTarget Property *1155*
ShowXGrid, TrendCmdTarget Property *1156*
ShowYGrid, TrendCmdTarget Property *1156*
ShowZoomBtn, TrendCmdTarget Property *1157*
StartNewFile , TrendCmdTarget Property *1157*
StatAverageValue, TrendCmdTarget Property *1158*
StatMaxValue, TrendCmdTarget Property *1158*
StatMinValue, TrendCmdTarget Property *1159*
TimeBrushColor, TrendCmdTarget Property *1159*
TimeBrushVisible, TrendCmdTarget Property *1160*
TimeFrameBtnColor, TrendCmdTarget Property *1160*
TimeScale, TrendCmdTarget Property *1161*
TimeTextColor, TrendCmdTarget Property *1161*
TrendBrushColor, TrendCmdTarget Property *1162*
TrendBrushVisible, TrendCmdTarget Property *1162*
TrendRunningType, TrendCmdTarget Property *1163*
VariableAddValue, TrendCmdTarget Property *1163*
VariableCursorPosIn, TrendCmdTarget Property *1164*
VariableCursorPosOut, TrendCmdTarget Property *1165*
VariableEnabling, TrendCmdTarget Property *1165*
VariableFreezedMode, TrendCmdTarget Property *1166*
VariableResetAllValues, TrendCmdTarget Property *1166*
VariableScrollEnd, TrendCmdTarget Property *1167*

VariableScrollNext, TrendCmdTarget Property *1168*
VariableScrollNextPage, TrendCmdTarget Property *1168*
VariableScrollPrev, TrendCmdTarget Property *1169*
VariableScrollPrevPage, TrendCmdTarget Property *1169*
VariableScrollStart, TrendCmdTarget Property *1170*
Vertical, TrendCmdTarget Property *1170*
ViewSamples, TrendCmdTarget Property *1171*
Visible, TrendCmdTarget Property *1171*
WeekBtnText, TrendCmdTarget Property *1172*
XGridColor, TrendCmdTarget Property *1173*
XGridLogarithmic, TrendCmdTarget Property *1173*
XGridUseNormalLine, TrendCmdTarget Property *1174*
XY, TrendCmdTarget Property *1174*
YearBtnText, TrendCmdTarget Property *1175*
YGridColor, TrendCmdTarget Property *1175*
YGridLogarithmic, TrendCmdTarget Property *1176*
YGridUseNormalLine, TrendCmdTarget Property *1176*
ZoomBtnText, TrendCmdTarget Property *1177*
ZoomMode, TrendCmdTarget Property *1177*

55. UIINTERFACE 1178

AlphaNumericEntry, UIInterface Function *1178*
ChooseColor, UIInterface Function *1178*
DoSomeEvents, UIInterface Function *1179*
EditRuntimeUsers, UIInterface Function *1179*
ExecuteCommand, UIInterface Function *1180*
GetLastActiveSynoptic, UIInterface Function *1185*
GetMonitorCoordinates, UIInterface Function *1185*
GetNumMonitors, UIInterface Function *1186*
GetPasswordFromLevel, UIInterface Function *1187*
GetPasswordFromUser, UIInterface Function *1187*
GetVariableNameFromList, UIInterface Function *1188*
GetWindowPos, UIInterface Function *1189*
HideLongOperationDialog, UIInterface Function *1189*
IsLongOperationAborted, UIInterface Function *1190*
LoadPicture, UIInterface Function *1190*
LogoffActiveUser, UIInterface Function *1191*
NumericEntry, UIInterface Function *1191*
OpenModalSynoptic, UIInterface Function *1192*
OpenSynoptic, UIInterface Function *1192*
OpenSynopticEx, UIInterface Function *1193*
OpenSynopticParameter, UIInterface Function *1194*
SayThis, UIInterface Function *1194*
SelectResourceFromList, UIInterface Function *1195*
SetDefPrinterOrient, UIInterface Function *1196*
SetRedraw, UIInterface Function *1196*
SetWindowPos, UIInterface Function *1197*
ShowHTMLDialog, UIInterface Function *1198*
ShowLongOperationDialog, UIInterface Function *1199*
ShowMenu, UIInterface Function *1199*
ActiveLanguage, UIInterface Property *1200*
MainVisible, UIInterface Property *1200*

56. USERANDGROUPCMDTARGET 1201

GetActiveUserObject, UserAndGroupCmdTarget Function *1201*
GetDesignGroupAtPos, UserAndGroupCmdTarget Function *1201*
GetGroupObject, UserAndGroupCmdTarget Function *1202*
GetNumDesignGroups, UserAndGroupCmdTarget Function *1203*
GetNumActiveUsers, UserAndGroupCmdTarget Function *1204*
GetNumRuntimeGroups, UserAndGroupCmdTarget Function *1204*
GetRuntimeGroupAtPos, UserAndGroupCmdTarget Function *1205*
GetUserObject, UserAndGroupCmdTarget Function *1206*
GetXMLSettings, UserAndGroupCmdTarget Function *1207*
LogonUser, UserAndGroupCmdTarget Function *1207*
ReadRuntimeUsersXML, UserAndGroupCmdTarget Function *1208*
SaveRuntimeUsersXML, UserAndGroupCmdTarget Function *1208*

AllowResizingForUsersBelowThisLevel, UserAndGroupCmdTarget Property *1210*
AllowRuntimeChangesForUsersBelowThisLevel, UserAndGroupCmdTarget Property *1210*
DefaultPrivAdminAccessLevel, UserAndGroupCmdTarget Property *1211*
DefaultPrivAdminLevel, UserAndGroupCmdTarget Property *1211*
DefaultPrivGuestAccessLevel, UserAndGroupCmdTarget Property *1212*
DefaultPrivGuestLevel, UserAndGroupCmdTarget Property *1213*
DefaultPrivUserAccessLevel, UserAndGroupCmdTarget Property *1214*
DefaultPrivUserLevel, UserAndGroupCmdTarget Property *1214*
EnableAutoLogoff, UserAndGroupCmdTarget Property *1215*
EnableNTUserLogin, UserAndGroupCmdTarget Property *1215*
EnableRuntimeUsers, UserAndGroupCmdTarget Property *1216*
MinimumPasswordLength, UserAndGroupCmdTarget Property *1217*
MinimumUserLength, UserAndGroupCmdTarget Property *1217*
SecsAutoLogoffTimeout, UserAndGroupCmdTarget Property *1218*

57. USERCMDTARGET 1219

GetDaysBeforePasswExpires, UserCmdTarget Function *1219*
GetGroupObject, UserCmdTarget Function *1220*
GetLastTimeUserAction, UserCmdTarget Function *1220*
GetListAccessVariable, UserCmdTarget Function *1221*
GetProp, UserCmdTarget Function *1222*
GetXMLSettings, UserCmdTarget Function *1222*
IsRemote, UserCmdTarget Function *1223*
ResetListAccessVariables, UserCmdTarget Function *1224*
SetProp, UserCmdTarget Function *1224*
AccessLevel, UserCmdTarget Property *1225*
AccountDisabled, UserCmdTarget Property *1226*
CannotChangePassword, UserCmdTarget Property *1227*
CommandListLogoff, UserCmdTarget Property *1227*
CommandListLogon, UserCmdTarget Property *1228*
Description, UserCmdTarget Property *1228*
Email, UserCmdTarget Property *1229*
EnableAutoLoggoff, UserCmdTarget Property *1230*
ExpiringDaysPassword, UserCmdTarget Property *1231*
FaxAreaCode, UserCmdTarget Property *1231*
FaxCountryCode, UserCmdTarget Property *1232*
FaxPhoneNumber, UserCmdTarget Property *1233*
Language, UserCmdTarget Property *1233*
Level, UserCmdTarget Property *1234*
Locked, UserCmdTarget Property *1235*
LogoffScript, UserCmdTarget Property *1235*
LogonScript, UserCmdTarget Property *1236*
MobileAreaCode, UserCmdTarget Property *1236*
MobileCountryCode, UserCmdTarget Property *1237*
MobilePhoneNumber, UserCmdTarget Property *1238*
MustChangedPasswordLogon, UserCmdTarget Property *1238*
Name, UserCmdTarget Property *1239*
OnLine, UserCmdTarget Property *1239*
Password, UserCmdTarget Property *1240*
SecsAutoLoggoffTimeout, UserCmdTarget Property *1241*
VoiceAreaCode, UserCmdTarget Property *1241*
VoiceCountryCode, UserCmdTarget Property *1242*
VoicePhoneNumber, UserCmdTarget Property *1243*
WebClientAutoLogoffSecs, UserCmdTarget Property *1243*

58. USERGROUPCMDTARGET 1245

GetNumUsers, UserGroupCmdTarget Function *1245*
GetUserAtPos, UserGroupCmdTarget Function *1246*
GetUserObject, UserGroupCmdTarget Function *1247*
GetXMLSettings, UserGroupCmdTarget Function *1248*
CommandListLogoff, UserGroupCmdTarget Property *1248*
CommandListLogon, UserGroupCmdTarget Property *1249*
DefaultAccessLevel, UserGroupCmdTarget Property *1250*
DefaultEnableAutoLoggoff, UserGroupCmdTarget Property *1250*
DefaultExpiringDaysPassword, UserGroupCmdTarget Property *1251*
DefaultLevel, UserGroupCmdTarget Property *1252*

DefaultLogoffScript, UserGroupCmdTarget Property *1252*
DefaultLogonScript, UserGroupCmdTarget Property *1253*
DefaultSecsAutoLoggoffTimeout, UserGroupCmdTarget Property *1254*
Description, UserGroupCmdTarget Property *1254*
Language, UserGroupCmdTarget Property *1255*
Name, UserGroupCmdTarget Property *1256*
WebClientAutoLogoffSecs, UserGroupCmdTarget Property *1256*

59. WORKSPACECMDTARGET 1258

OpenProject, WorkspaceCmdTarget Function *1258*
OpenScreen, WorkspaceCmdTarget Function *1258*
OpenScript, WorkspaceCmdTarget Function *1258*

1. Preface

1.1. Introduction

Programming the Movicon VBA Script Language guide contents.

The guide contents of the programming Basic Scripts VBA™ compatible (Visual Basic for Applications™) contain all the information the developer needs to know for realizing Movicon 'Basic Script' routines.
One part of the guide is dedicated to the instructions inherent to specific commands for Movicon, and another part contains the standard VBA™ compatible instruction syntax, which form part of the **"WinWrap Basic Language"** guide incorporated in Movicon.

1.1.1. Preface

All the information contained in the Movicon documentation is based on the assumption that:

- Windows 32/64 bit versions refer to the Microsoft inc. trade mark
- Movicon refers to the supervision system developed by Progea and is protected by the international Copyright
- VBA is referred to the Microsoft Visual Basic for Application
- Ms Access and SQL Server refer to the Microsoft inc. trade mark
- Any other product or brand mentioned is covered by Copyright on behalf of its owner

2. General Concepts

2.1. Basic Scripts in Projects

You can use Basic Script routines inside projects in different circumstances and modalities.
It is best to use them in situations when the same operations can not be done with other resources or methods: general logic, drawing execution properties, events. The unconditional use of the basic scripts in projects is very handy in the project design time phase but may slow down project execution and consume more of the project's resources.
The basic code can be used in more areas of the project: as resource, directly in the execution properties of drawing objects, as codes associated to the events of objects (alarms, drawings or symbols), in screens.
These functionalities are discussed in more detail in the relating sections.

2.1.1. WinWrap Basic Language

Movicon has taken on board a software component called WinWrap Basic which allows routines in Basic language to be integrated into applications.
Listed below are some of the features and things WinWrap Basic lets you do:

- create routines in VBA language (Visual Basic for Application) compatible with the BASIC code (Beginners All-purpose Symbolic Instruction Code)
- extend instructions sets with customized functions methods
- create dialog boxes by using the Windows standard controls (buttons, checkboxes, groupboxes, listboxes, option buttons, images and text boxes)
- declare and call the Windows API AND WMI functions
- supports ActiveX controls

2.2. VB.NET and Unicode Support

Support to VB.NET

By using the special **'#Language "WWB.NET"** key you can pass over to the .NET programming mode. This consents you to access directly to the .NET assemblies with the VB. code. The dialog window used for adding reference objects to be used in code considers this key. In cases when .NET language usage has been set, you will be able to view the list of all the assemblies which can be selected.

Code example:

```
'#Reference #System.Windows.Forms, Version=2.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089, processorArchitecture=MSIL

'#Language "WWB.NET"

Dim WithEvents t As System.Windows.Forms.Timer

Sub Main
        t = New System.Windows.Forms.Timer
        t.Interval = 1000
        t.Enabled = True
        Wait 1
```

```
End Sub

Private Sub t_Tick(ByVal sender As Object, ByVal e As System.EventArgs) Handles
t.Tick
        Debug.Print Now
End Sub
```

A series of new instructions to render the code compactible to VB.NET programming has been provided especially for this purpose.

Restrictions:

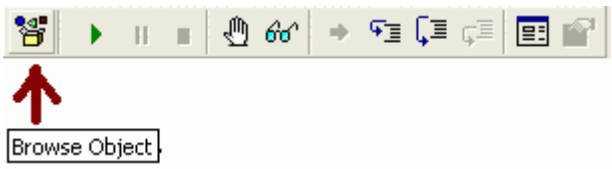
- .NET programming is not supported in Windows CE
- The Subroutines in objects cannot be renamed as would be consented in .NET programming
- Not more than one event can be linked to the same procedure using a list of "Handles"
- Variable Script Events cannot be managed in WWB:NET scripts
- In WinWrap Basic version 9, the basic script is single thread and the script code must be executed from the thread that created it. For example, a .NET delegate can not be passed to an object that manages this delegate in another thread. A good example of this would be some of the "System.IO.FileSystemWatcher" class functions that create additional threads for monitoring file modifications. In this case the methods of this class will not support other thread calls. This is also the case for the "System.Net.Sockets" class.
- In WinWrap Basic version 9, the basic script do not support nested types, being a type defined in the ambient of another type. For example, it is not possible to use the "System.Net.WebRequestMethods.Ftp" functions.

Unicode Support

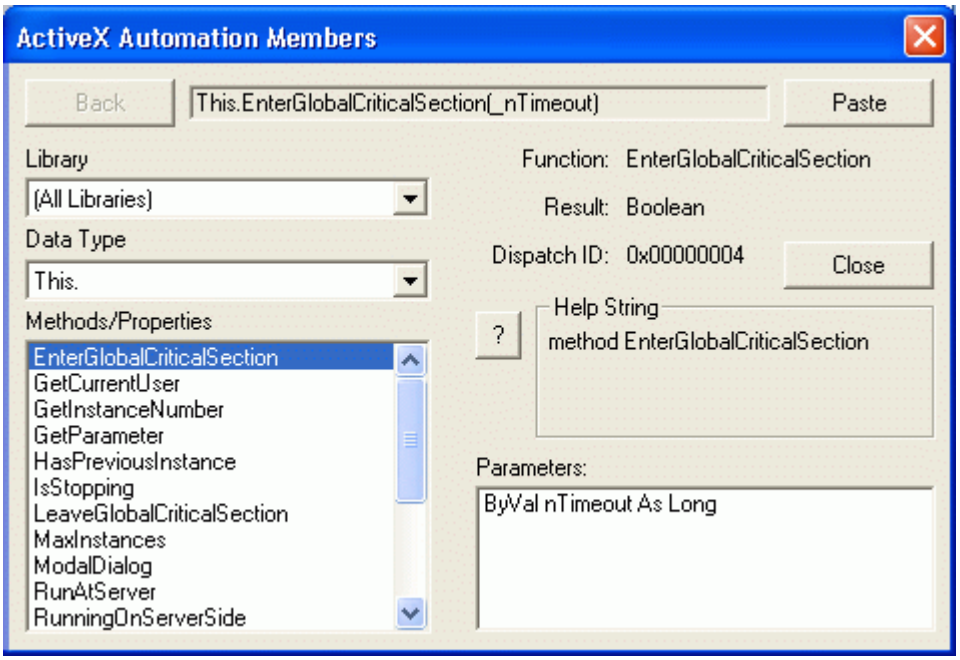
The code editor consents you to insert strings in Unicode format. Therefore you can view the Unicode strings in the basic script dialog window or assign Unicode texts to the project's string variables.
UTF8 or UTF16 Unicode files can be read and/or written by writing one of the two new "vbUTF8BOM" and "vbUTF16BOM" constants, added for this purpose, in the first character of a text file in order to determine its code.

2.2.1. Basic Script Libraries

In addition to the basic functions provided by WinWrap Basic, you can use a series of supplementary functions inside the Basic Scripts which belong to the Movicon libraries and permit project interaction. These functions permit you, for example, to read and write Movicon Real Time DB variables, to execute page changes, to interact with the Movicon symbol properties, and much more. The Movicon function libraries are called Basic Script Interface Libraries, and each interface has a collection of sets of specified functions for specific project components.
There are a vast number of Movicon basic interfaces and the list of functions in these libraries can be accessed with the "Browse" button from the basic tool bar (which is available after the a basic code, of any component, has been opened):



The window which opens shows the name of the interface in the "Data Type" box and the methods and properties relating to that interface are listed in the "Methods/Properties" list box:



Another interface can be selected by using the "Data Type" list box.
To get a function's help just select the function and click the question mark if found in the functions' Browse window, or if inside a basic code just select the function and click the F1 key. The various fields in the Browse window mean:

Back

Returns one step back on the libraries hierarchy scale. Some libraries are set with objects which methods and properties are associated to, for instance when displaying the property of an object and clicking on the back button should return you back to the list of objects set in the library.

Paste

Copies the contents of the box at the side, in the point where the cursor is situated in Script's editor window. If the command is disabled indication will be given by telling you it is impossible to copy the contents in the position indicated by the cursor.

Close

Closes the browse window and the Script's editor returns active.

Library

Allows you to select one of the library proposed by the list. When ActiveX/OCX have been added by using the references, these will also be on the list.

Data Type

Allows you to select a data type from the list proposed. The list refers to the library selected in the Library box.

Methods / Properties

Allows you to select a method or a property from the list proposed. The list refers to the data type selected in the Data Type box.

?

Accesses the help of the property or method selected in the box at the side. Some external libraries, not setup by Progea and enabled through the references, do not install their help files.

Parameters

This displays any eventual list of parameters set for the method selected in the Data Type box.

2.2.2. Subs, Functions, Events, Methods and Properties

Subs and Functions

Routines can be created in basic scripts which are basically portions of code enclosed in a block and come in two types:

- Sub
- Function

The difference between these two types or routines is very slight. Both can be called by parameter passing. The difference of the "Function" compared to the "Sub" is that is can return a value type set by the programmer (Bool, Int, String, etc.).

Example 1:
The Sub Test is called inside the Sub Main of a Basic Script:

```
Sub Main
        Call Test()
End Sub

Sub Test()
        MsgBox("Test Sub", vbInformation + vbOkOnly,GetProjectTitle)
End Sub
```

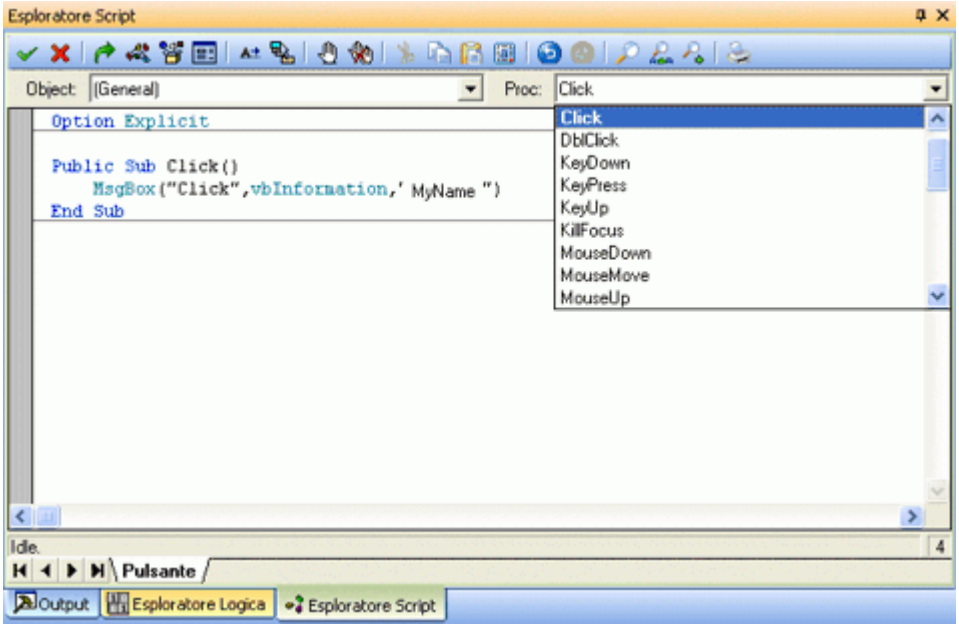
Example 2:
The Function Test is called inside the Sub Main of a Script and the key pressed by the user in the MsgBox is put on log:

```
Sub Main
        Debug.Print Test()
End Sub

Function Test() As String
        If MsgBox("PTest Function",vbInformation + vbOkCancel,GetProjectTitle) = vbOK
        Then
                Test = "OK"
        Else
                Test = "Cancel"
        End If
End Function
```

The routines described above can be called by other routines, creating nested calls between them. There must always be a **"Sub Main"** in the Basic Script Resource which is the routine automatically executed by the Basic Script when run. However, it is the programmer's job to insert the right codes inside this routine and call any other Sub or Function they themselves have created. Once the instructions contained in the Sub Main() have all been executed the basic script is ended and must be called again in order to be run another time. In any case loops can be inserted inside the Sub Main() to keep the Basic Script always running.

Events

There are Sub routines provided by the system (Movicon or any inserted ActiveX component) which are automatically called by the system at the forefront of certain events. These routines, which are actually called "Events", can be inserted in the script and it is up to the programmer to add the desired codes inside them.



Example:
When the following code is inserted inside the a rectangle design's script, a MsgBox with the word "Click" will appear every time the rectangle is clicked on with the mouse:

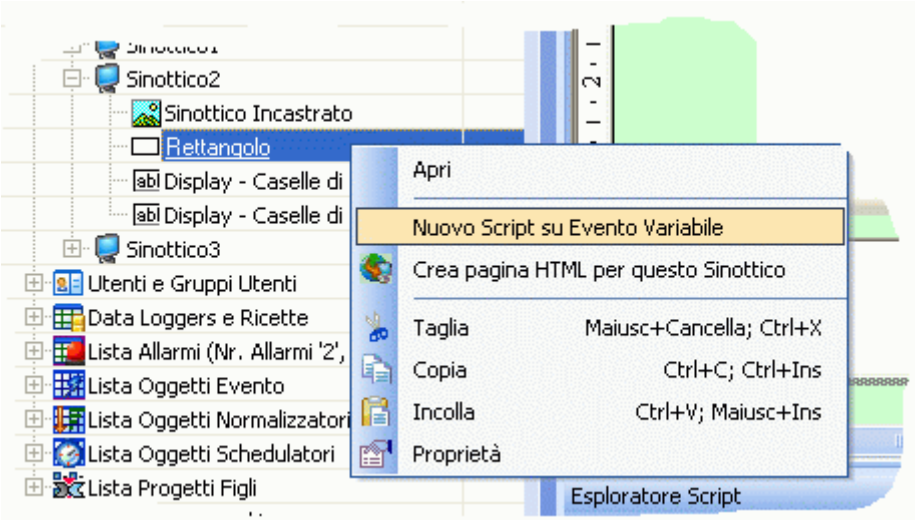
```
Public Sub Click()
        MsgBox("Click",vbInformation,GetProjectTitle)
End Sub
```

In this case the Public Sub Click() is an event triggered by the system when the rectangle is clicked, while the code within has to be inserted by the programmer.

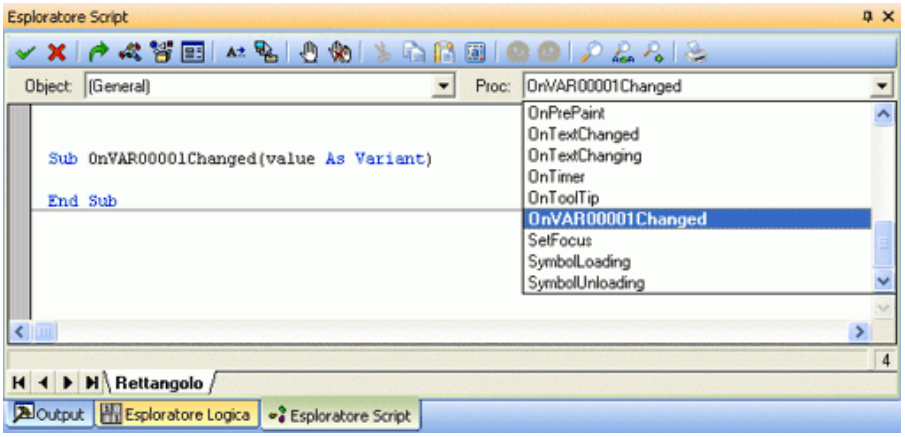
A series of events provided by the system can be selected within the basic codes of symbols, screens and alarms. These event exclusively concern the component in which the code is being edited, meaning the events in question which regard events triggered by the component. These events are listed in the "Proc:" list when the "(General)" item is selected from the "Object" list, as shown in the figure above.

Customized events on specific variables from the Movicon Real Time DB can be created for the Symbol and Drawing category and also for Screens.
For instance an event can be inserted to be called each time a certain project variable changes state. This procedure is very handy for keeping status changes monitored without overworking the system. The procedure for creating an event of this type is as follows:

- Right click on the component or the screen you wish to associate the event to from the "Project Explorer" window to open the menu from which you must select the "Add New Variable Script Event" command:



- This will open a Browser window of the project's variables from which the variable of interest is to be selected. Once selected the variable inside the script code of the component or screen will be automatically inserted in a new event on the "Proc:" list called "OnVariableNameChanged" (On + variable name + Changed. For example when the VAR00001 variable is added, the event will become "OnVAR00001Changed").



When inserting this new event into the code it will be called each time the value of the "VAR00001" changes and the "value" parameter will return the actual value of the variable. As always, more than one variable can be associated to each component.

Caution! The basic interface events are not executed when containing spaces between the sub's name and at the beginning of the parameter.
For instance:
Public Sub OnSetAlarm (bSet As Boolean, bRet As Boolean) -> will not be executed
Public Sub OnSetAlarm(bSet As Boolean, bRet As Boolean) -> will be executed

Methods and Properties

The functions which can be inserted from the Movicon basic interfaces (or from other external libraries such as ActiveX/OCX) can be defined as "Methods" and "Properties". The properties identify the characteristics of the object in question, ie. color, title etc, while the method executes the function of that object.

2.2.3. Variables in Basic Scripts

In the Movicon Basic Scripts you can use both Movicon Real Time DB variables, which result as global variables for all the project, and Basic Script local variables which are only visible within Basic routines and which are destroyed once the Basic Script has terminated.
The project's variables (Movicon Real Time DB) can be directly used by their names or by using the "DBVariableCmdTarget" (GetVariableValue(), SetVariableValue()) functions. In cases where structure variables are used directly, the (":") colon character used for separating the variable name and the member name is no longer used and can be replaced with the ("_") underscore or (".") point characters. However, the colon character can still be used if enclosed within square brackets ([Structure:Member]). The standard colon character is used in all the basic functions which require a variable as parameter (ie. GetVariableValue(), SetVariableValue(), etc.).

It is not possible to directly use the name of variables declared in the project's Real Time DB which start with a number or an underscore ("_") character. The reason for this is due to the fact that the VBA interpreter does not permit variable declarations of this type and therefore in cases such as these it is necessary to always use the GetVariableValue(), SetVariableValue()) functions.

The variables internal Basic Scripts have priority over those of the project's. Meaning that when using the name of a project variable directly in the basic script code where a variable already exists with the same name, this will be ignored and the already existing one will be set or read. In this case you will have to use the GetVariableValue(), SetVariableValue() basic functions.

Bit type variables, when used directly with their name, are converted into boolean (true or false). This means that a project variable with value "1" will be read as a "true" variable inside the Basic Script and therefore with a numeric value equal to "-1". The "0" value is interpreted as "false" by the Basic Script which will always be a numeric value equal to "0". This mechanism also goes for both the reading and writing of bit variables inside Basic Scripts. The GetVariableValue() and SetVariableValue() basic script functions can always be used to obviate this mechanism. In this way the function's return value will always be the numeric value of the bit, "1" and "0", and not the boolean value.

Example:
When reading the bTest variable declared in the Movicon DataBase as bit and set to "1", the result will be:

```
Sub Main()
        Debug.Print bTest 'Result = -1
        Debug.Print GetVariableValue("bTest") 'Result = 1
End Sub
```

IntelliSense use for RealTimeDB variables

The VBA IntelliSense can be populated by the project's global variables (the RealTimeDB variables). This way you can get a variable using the drop-down list that appears pressing the "Ctrl+Spacebar" and have direct access to structure's members and bytes array's elements.
Among variable properties a special option has been added, called "IntelliSense", which can be used to add the variable to the VBA IntelliSense list. The default value for this option is "disabled", but it can be enabled for all variables the user wants to be available in the VBA IntelliSense.

Bytes Array's elements and members of Structure variables, defined in the project's RealTimeDB, can be accessed (if the "IntelliSense" property has been enabled) within the basic script code by using the following syntax:

ArrayVarName.eMemberIndex (where MemberIndex starts from "0" to "array size -1". Ie. "Motor.e5" indicates the sixth "Motor" array element)
StructureVarName.MemberName

LIMITS: the IntelliSense does not show the screen local variables and is not populated with global variables if used in "Dynamic Properties Explorer".

We suggest to enable the "IntelliSense" option only for those variables you intend to use in VBA scripts. The operation of populating the IntelliSense while opening or executing a script can be slow if the number of variables with this option enabled is large (i.e. thousands).

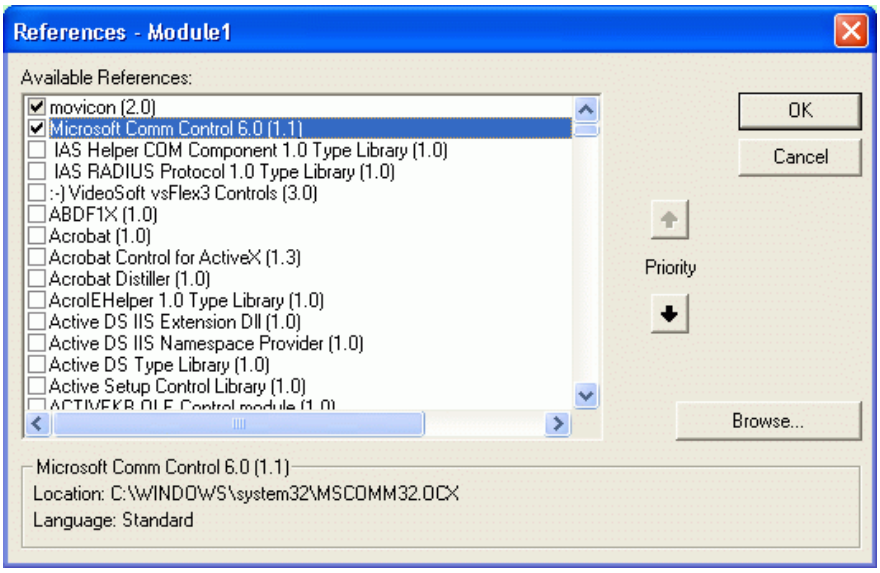
Variables without sign

The WinWrap does not provide the use of variables without sign therefore in order to use a RealTimeDB WORD variable type (without sign) in the Basic Script it must be converted to avoid causing any overflow errors. Overflow error occur when the variable used in the script exceeds the INT type value (ie. 32767 for one Word) because the basic script engine does not manage variables without sign. **To avoid this problem, a WORD type variable is always passed to the WinWrap as a Long type and a DWORD type variable is passed to the WinWrap as a "Hauge_" type which means a "64 bit integer value".**

WORD type variables can be managed in the WinCE environment but not DWORD type (when exceeding a Long value) as WinCE does not support "VT_18" data types.

2.2.4. Quick Programming

Some keys can be used for inserting functions quickly while writing a script code. For instance, when the "CTRL+Space" keys are pressed a drop down list will appear with all the functions (WindWrap Basic or Movicon) which can be inserted in the context of that script.
The same thing can be done when using Object type variables where a series of methods and properties can be made available by just writing the Object's name followd by "." (dot) to see the list of these functions. When using the ActiveX/OCX, this can only work when the library containing the control in question has been enabled in the basic's references and when the Object has been declared with the name of the class to be implemented and not as a generic "Object". For instance, if you want to access the property of an ActiveX MsComm, you need to enable the “Microsoft Comm Control 6.0” reference.

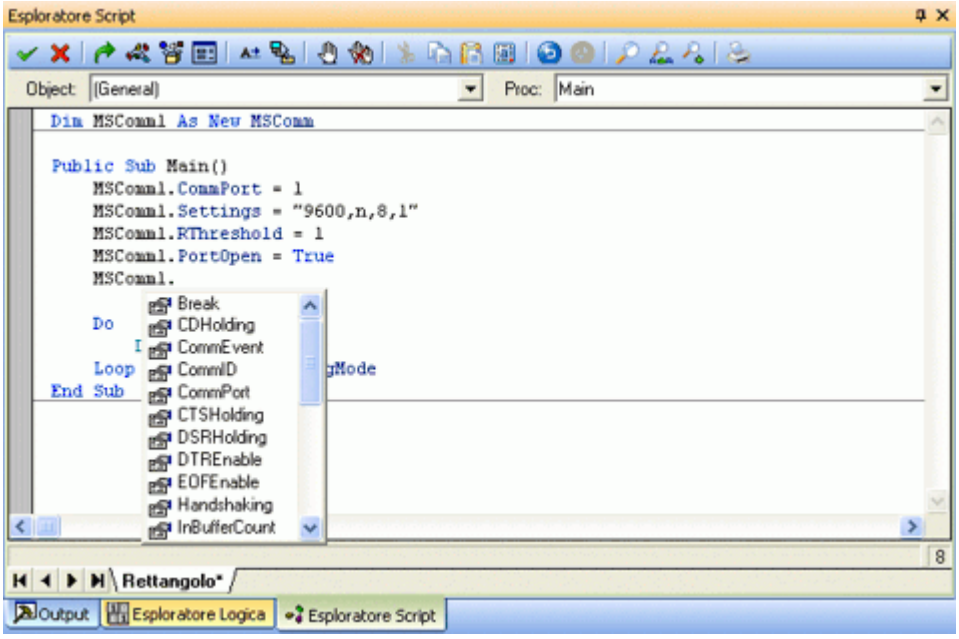


Then you have to declare the object inside the Basic Script as follows:

```
Dim MSComm1 As New MSComm

Public Sub Main()
        MSComm1.CommPort = 1
        MSComm1.Settings = "9600,n,8,1"
        MSComm1.RThreshold = 1
```

```
            MSComm1.PortOpen = True

            Do
                DoEvents
            Loop Until IsInStoppingMode
        End Sub
```

when declaring the MSComm1 variable as shown above, which means not as Object but as MSComm (ActiveX class to be used), by writing the Object name followed by a dot, the following window will display:



For further information on using ActiveX/OCX inside Basic Scripts please refer to the paragraph headed "ActiveX/OCX in Basic Scripts".

2.3. Code in Basic Scripts and in Symbols

There is a fundamental difference between the functioning of the Basic Script resources and the codes which are inserted inside symbols, screens and alarms. The **"Sub Main"** must exist in Basic Scripts which is executed when called by a basic script. However,the "Sub Main" does not exist in symbols (or screens or alarms), where codes are managed exclusively based on events (SymbolLoading, Click, etc.) which are available in the codes and which were inserted by the programmer. Movicon calls this events only when the symbol (or screen) is active and therefore when the screen is loaded into ram.

2.3.1. Basic Scripts as Resources

When you want to insert a Basic Script into the project, you need to use the inserting new resource procedures. To insert a new Basic Script resource first select the "Basic Script" group from the tree structure shown in the "Project Explorer" window, then right click to open a menu and then select the "Add new script" command.

This operation is confirmed by the appearance of the new Basic Script resource in the group or the point selected in the Resource structure along with the opening of its code editor window. At this point you can enter the VBA™ code as described in the paragraphs specifically written for this topic.
The resource can then be assigned a Name by using clicking on it and typing in the name to replace the temporary one.

A Basic Script resource must contain the Main (Sub Main) procedure inside. The instructions contained in this subroutine will be executed when the basic script is launched from the project's logic. At the end of the subroutine, without any programmed loop cycles, the basic is terminated and made ready for the next call.
The Main procedure does not present any configurable parameters but they can be associated when the Basic Script is called. The parameters should be indicated in the command which follows the basic script call where each one is divided by a comma and the GetParameter(), internal the basic script, can be used for reading the values of the last parameters with which the basic was called.

After a Basic Script resource has been put into execution for the first time, even after the Sub Main() has stopped, the resource will remain active, therefore any events which have been set in this resource will be executed on occurrence. For instance, an "Event on Variable" can be inserted in the Basic Script which will be executed upon variable change even when the Basic Script has already stopped the Sub Main(). On the other hand, if you want to stop the execution of a Basic Script completely, you need to use the "Stop" command which can be found in the Movicon "Command List". The two Basic Script's "Loading" and "Unloading" events are called respectively at the Script's first start and stop (by using the 'Stop' commands or when the project is closed).

The stop command of a basic script resource unloads only those basic resources which are being run in separate threads from memory. The other basic script resources being run in the same thread are only stopped. As a consequence of this, the "Unloading" event is no longer executed following a stop command for those basic script resources which are not in separate threads. In addition to this the variables used by the basic script always remain in use once the basic script has been run at least once.

A Basic Script resource introduced into the project can be put into execution in various ways according to what is required. The following paragraphs illustrate these methodologies.

Execute on command

The execute on command is easily set through the "Command List" properties of the objects, menus or resources which are provided with the possibility to execute commands.
For example, when you wish to associate the execution of a Basic Script routine to a button, you need to select the "Script Commands" from the "Commands" properties, and select the Basic resource desired from the proposed list referring to the ones introduced, and then define the other settings as desired.
The execution on command of a Basic Script can be also done from the "Command on Event" resource, where the basic routine activation will not be managed by a command from the operator but by a specific event.

Execute at startup

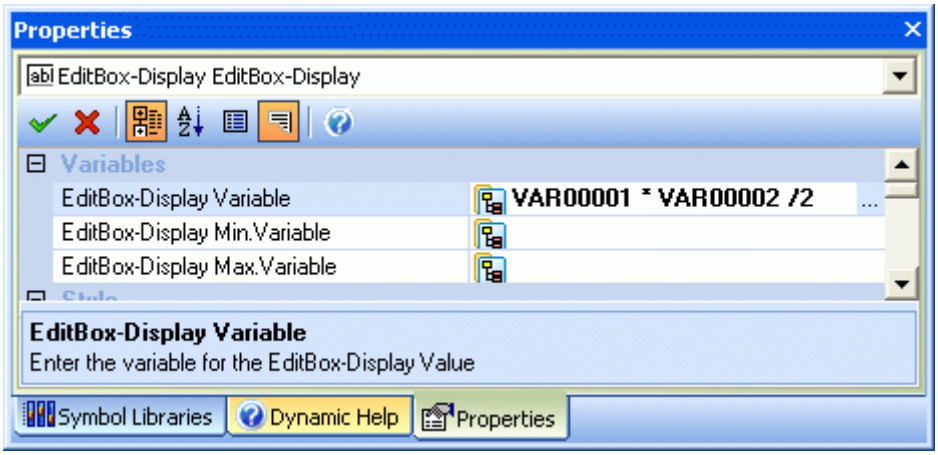
You can execute Basic Script routines automatically at the startup of the applied project in Runtime. To use this function, you need to access the appropriate "Startup Script" setup from the "Project Execution Settings".
By using the right selection window you need to select the Basic Script resource from those previously inserted into the project.

Execute from another basic script

The execution of a basic script can be done from another Basic Script with the "RunScript" function. When both the basic scripts, the one containing the function and one being launched, are not in separate threads, the basic script's execution will be buffered and executed as soon as the chance arises.

2.3.2. Basic Script Expressions in Object properties

The animation characteristic settings available in Movicon objects (Displays, Rectangles, etc.) allow you to establish the variable and the relevant thresholds for activating the graphical functions. Movicon allows you to exploit a particular characteristic for activating the graphic functionalities: the association of a Basic Script string expression for command execution.
By using this concept, instead of a variable you can associate a on whole VBA expression, complete with variables and logic conditions, to execute the function desired. By doing this the value returned from the expression will be used instead of the variable value.

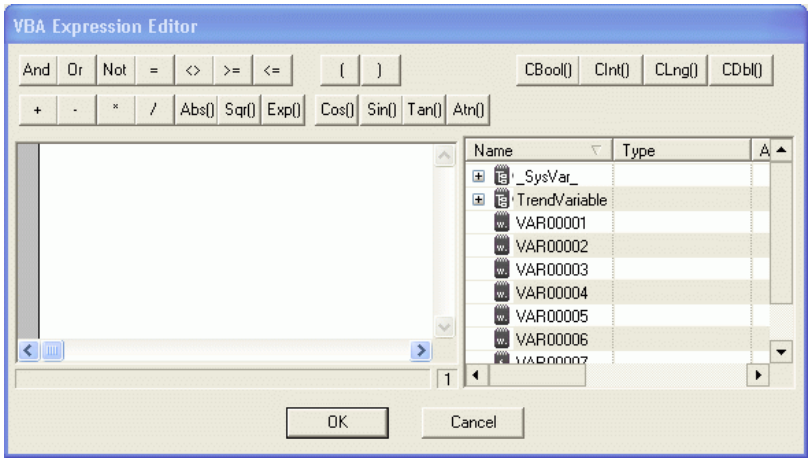


This illustration shows how you can insert a Basic Script expression instead of a variable, The value returned from the expression will be interpreted instead of the variable's.

Caution! Movicon script functions such as GetVariableValue(), SetVariableValue, QualityOf(), etc., cannot be used in basic expressions. The basic expressions only support WinWrap mathematical operators as listed below.

VBA Expression Editor

The basic script expressions can be inserted directly instead of the variable. An editor window has also been provided to help the programmer to do this and which can be opened with the "Expression..." button found in the "Tag Browser" window:



This editor window is very handy as it provides all the operations needed for inserting expressions allowing major control over any errors.
Once the window has been opened for inserting a variable double click on the variable name from the list to the right. To insert a mathematical function use the corresponding button. The available functions are:

Operator	Description
And	AND binary operator
Or	OR binary operator
Not	NOT logic operator (inverts the variable's boolean value or the expression which follows it)
=	equal to binary operator
<>	different than binary operator
>=	more than or equal to binary operator
<=	less than or equal to binary operator
(	open brackets
)	close brackets
+	addition
-	subtraction
*	multiplication
/	division
Abs()	absolute value (module)
Sqr()	Square root
Exp()	base power is (2,718)
Cos()	cosine (in radiants)
Sin()	sine (in radiants)
Tan()	tangents (in radiants)
Atn()	arctangent
CBool()	conversion to boolean (boolean)
CInt()	conversion to integer (integer)
CLng()	conversion to long (long)
CDbl()	conversion to decimal number with double precision (double)

Long expressions can be inserted by going to the beginning with the Enter key. When the expression inserted is not valid, an error message will show when confirming with the OK key.

Expressions may not be valid if producing a division by zero. When validating expressions, the variables are considered with the "1" value. Therefore, the "VAR00001/(VAR00002-1)" expression will not be validated by the expression editor because it produces a division by zero.

Using the VBA expressions inside objects is quite fast even under Windows CE, and should be considered that it require less resources than writing code inside object events. Infact, just only one thread is related to manage all the VBA expressions for the objects loaded into the memory, and these expressions are evaluated only when on change (event) of the related variable. This can be faster also than the IL Logic, because the VBA expression are evaluated on event, instead the IL Logic is executed continuously in a cycle.

This functionality allows you to extend the drawing's graphical potentialities further. The variable associated to each one of the symbol or drawing graphic properties can in fact be substituted with Basic expressions which can contain combinations of more variables or logic or mathematical expressions.

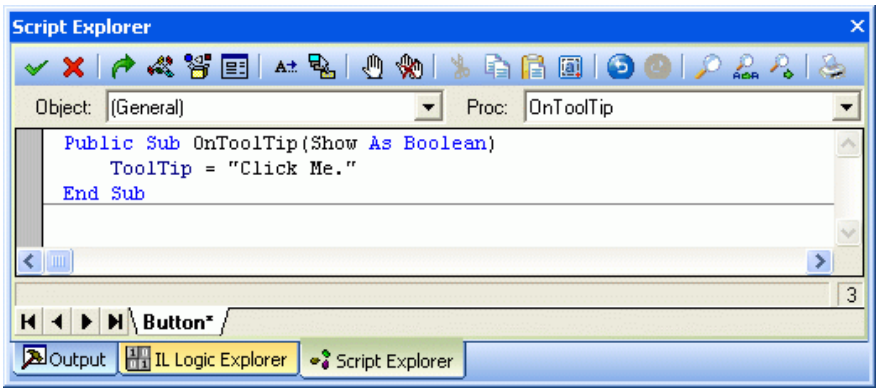
When a symbol containing a VBA expression is added on screen, i.e. “VAR00001 + VAR00002”, Movicon acknowledges the use of the two “VAR00001 and VAR00002” variables and inserts them both in the project.

Caution! This functionality passes the expression inserted directly in the Basic Script interface, therefore the use of variables that begin with numeric characters or particular characters that include an underscore ("_") will not be admitted.

Caution! Basic expressions do not support decimal formats (i.e. "x.x") but only "x". Therefore you should used decimal order divisions directly in the expression (es: (VAR00001+VAR00002)/10).

2.3.3. VBA™ Basic Script in Object and Alarm Code

The thresholds of on-screen alarms, drawings or symbols and screens can contain script codes. The managing of these basic codes inserted into objects is different from that of the Basic Script resource, whose execution is controlled in the project's logic. A series of events have been provided internal the project which, when selected and then inserted inside the basic script editor, allow you to insert codes. When these events are generated while the project is being run, the basic routines in these events will be executed.
Each object may have different properties, methods and events according its function type. Please refer to the lists in the specific paragraphs to get further details on these.



It is important to remember that the code associated to an event of an object is executed only when the object is loaded into memory and then managed by Movicon. For instance, a drawing associated with a code is executed only when the screen container is loaded.

Please keep in mind that the codes inside drawings are not initialized straight away when the screen page is loaded but only when needed. If a symbol contains the "SymbolLoading" event Movicon is obligated to initialize the basic script code contained in that drawing straight away. This means that page loading is quicker when the drawings associated to it do not contain the "SymbolLoading" event. However this does not mean that the "SymbolLoading" event should not be used altogether but only when necessary. This should be taken into consideration particularly when creating Templates, above all when they are to be used in Windows CE.

The script codes of drawings are loaded only when they are needed and not when the page is being loaded.

Script Code in Alarm Thresholds

Script code can also be associated to each alarm threshold. This is done by simply selecting the threshold with the mouse and using the "Script Explorer" window. A series of events are also available in the alarm threshold code which when selected and then inserted in the basic script editor will allow you to insert code. These events include for example: "AlarmLoading", "OnSetAlarm", etc. For further information please refer to the related section in the "AlarmThresholdCmdTarget" scrpt interface. Since a few restrictions exist in using script, they should only be used in alarms when really necessary, one reason being is that they do not get managed in redundancy. The other reasons causing restrictions of use can be found in the chapter on "Alarm Managment" from the Movicon "Techical Specifications".

2.4. Public Basic Scripts

When single functions, subroutines or procedures of the basic script code being used is needed in other parts of the project you can make it 'Public' so that its contents can be called by other script codes. This means that you do not have to write the same script code repeatedly. Therefore any other modifications need only to be done once in the editor of that resource saving you a great deal of time.

In order to access from a Basic Script to the functions of another Basic Script you need to use the "Uses" function. This function allows you to implement, internal the Basic Script, the functions contained in the specified Basic Script. The complete syntax is as follows:

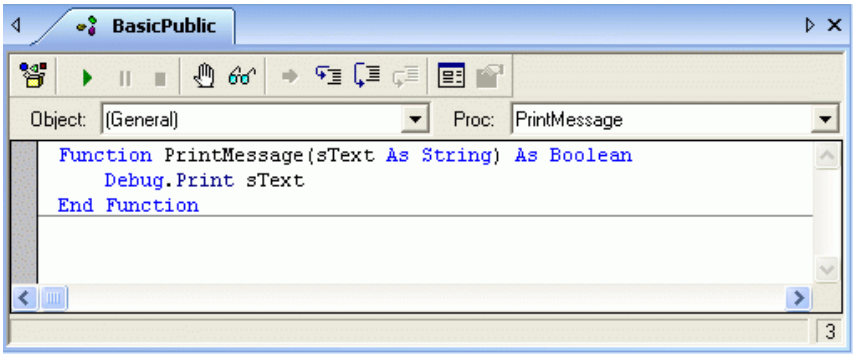
'#Uses "*BasicScriptName"

The "*" character before the name of the basic routine allows Movicon to interpret the XML code contained in the Basic Script file.

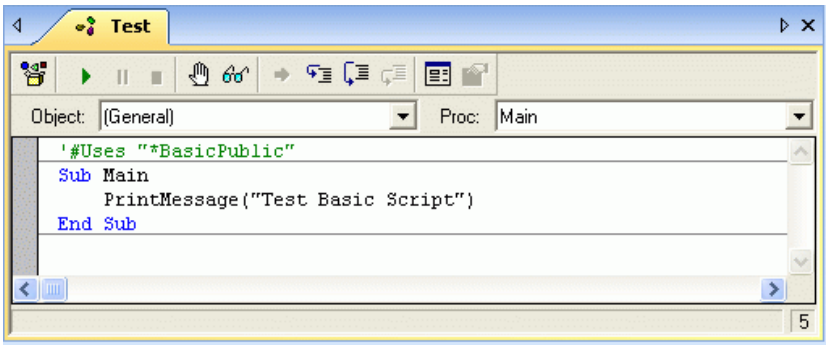
IMPORTANT! The '#Uses' instruction in the symbols code can only be used for functions, subroutines, public vars or public constants. It cannot contain the symbol's events as the actual managing of the code's"delay load" does not allow it to be known that there are events to be processed for the symbol in the shared basic script.

The "Open #Uses" command is found in the "Script Explorer" window or the "Edit Menu (Basic Scripts)" and consents the availability of functions and objects declared in the scripts "#Uses" modules to the IntelliSense (with the CTRL + Space command). When using the CTRl + Space command a list will display showing all the available functions including both the standard script functions and those defined in the #Uses module.
A module called through the "#Uses" instruction can call another module where the "Open #Uses" command can be reapplied to make those functions it contains available to the IntelliSense as well.

Example:
Let's suppose we have to create a basic script called "BasicPublic", within which the "PrintMessage()" function has been inserted. The"PrintMessage()" function prints the text passed as a parameter:



At this point when you want to call the "PrintMessage()" function from another basic routine, or also from a script code of a drawing or screen, etc. you only need to use the syntax described above inside the script. Let's go on and suppose we want to implement the function in the "Test" script, the code would then be as follows:



IMPORTANT! When a Basic Script uses a function from another Basic Script, not only the function being used but also all the code of the public Basic Script will be implemented in its internal. Therefore functions, routines or constants with the same name cannot be declared in both Basic Scripts.

More Basic Script routines can be created to be shared in the same project. This permits you to structure and manage any public functions in groups to be used in the project.

Nested Public Basic Scripts

Public Basic Scripts can also be managed in tree structures, meaning that a Basic Script can share another which is sharing yet another one. However in this configuration each Basic Script can use only the routines, constants or variables from the Basic Script being shared with.
For instance, lets suppose a BasicScript1 has been created which shares BasicScript2 and that Basic Script2 shares BasicScript3. In this situation BasicScript1 can call BasicScript2's routine, but not BasicScript3's, which instead is used by BasicScript2.

Public Basic Scripts between Parent and Child projects

The Basic Script resources can also be shared between Parent and Child projects. In this case the following syntax is to be used:

Sharing a Basic Script resource in a child project by the parent project:

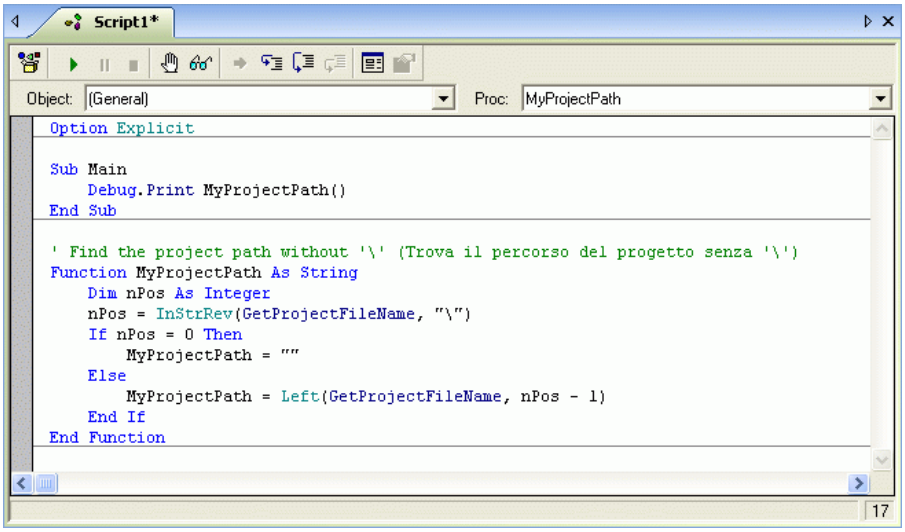
'#Uses "*ChildProjectName\BasicScriptName"

Sharing a Basic Script resource in the parent project by the child project:

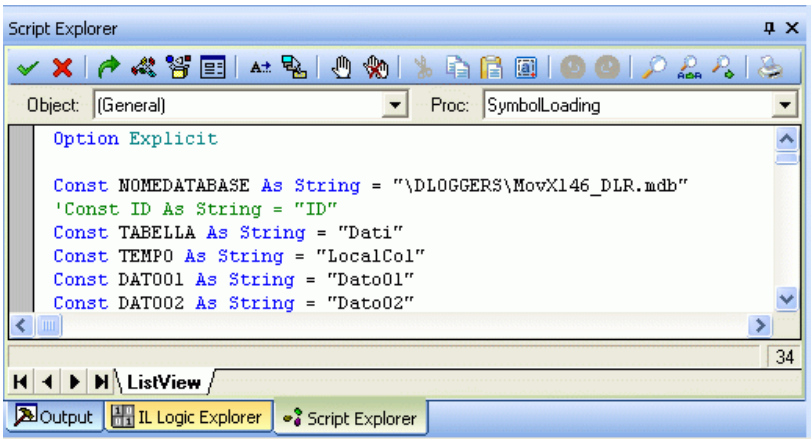
'#Uses "*..\BasicScriptName"

2.5. Basic Script Editor

The basic script editor is composed of a window containing an area with a white background in which you can edit codes, plus a series of commands as described below.
The window can be resized and the possibility to open more than one window at the same time also permits you to quickly carry out Copy&Paste operations from different projects. To open the editor just click on the Basic Script resource.



The script code editing of objects in screens and alarms, however, is done through the Movicon **"Script Explorer"** window. The contents of this window change dynamically based on the component selected, by show its associated script code.



Both the editing windows shown above provide two drop-down lists, **"Object:"** and **"Proc:"**.

The **"Object:"** list displays a list of basic set objects in Movicon which correspond to **"(General)"** and **"On_Events"**. The selection of one object in respect to another changes the list of procedures in the **"Proc:"** list available for that specific object.
The **"(General)"** object identifies the procedures and events programmed for that specific object and the list can change according to where the code is being edited: Basic Script, Drawing, Screen or Alarms Resource.
The **"On_Events"** object identifies the general Movicon events and is available only when the code associated to the project is edited.
Other item can be found in the **"Object:"** list which correspond to eventual Active/OCX components inserted in the basic code and which provide their own events.

The list of available events based on the selected made in the **"Object:"** list is displayed in the **"Proc"** list.

2.5.1. Edit Menu (Basic Scripts)

The opening of a Edit window of a Basic Script code shows a menu of a set of specific commands for editing the code as well as the classic Windows commands:

Undo (Ctrl+Z; Alt+Backspace)
This cancels the last operation carried out (ie. cancellation of a resource, changes to an object, etc).
This command is also available in the Movicon **"Tool Bar"**.

Redo
This restores the last cancelled operation (ie. cancellation of a resource, changes to an object, etc).
This command is also available in the Movicon **"Tool Bar"**.

Cut (Ctrl+X; Caps+Canc)
This cuts the object, the resource or anything else which has been selected. In this case the object is cancelled by kept in memory on the Windows clipboard.
This command is also available in the Movicon **"Tool Bar"**.

Copy (Ctrl+C; Ctrl+Ins)
This copies the object, the resource or anything else which has been selected in memory on the Windows clipboard.
This command is also available in the Movicon **"Tool Bar"**.

Paste (Ctrl+V; Caps+Ins)
This pastes the object, the resource or anything else, which has been previously Copied or Cut onto the Windows clipboard, on the area of the workspace selected with the mouse.

This command is also available in the Movicon **"Tool Bar"**.

Delete (Canc)

This deletes the selected object without copying it onto the Windows Clipboard.

Find what (ALT+F3)

This command is available when a preset resource has been opened and lets you specify the character or text string to be searched for within the resource selected.
According to the standards used it might be necessary to specify whether the text to be searched for must respect the Upper/Lower key characters.
The Next button starts a new search, while the Cancel button ends the search.

The Find command can also be accessed from the keyboard with ALT+F3, or, if available, from the **"Tool Bar"** through the Find edit box.
This command is also available in the Movicon **"Tool Bar"**.

Find Next (F3)

After having started a search with the Find command, you can search for the next specified text with the Find Next command which can also be executed with the F3 key.
This command is also available in the Movicon **"Tool Bar"**.

Replace

The Replace command allows you to specify a text to be searched for and a text to replace the one found.

Select All

The select All command allows the simultaneous selection of all the resource contents currently active, when consented.

Font

Sets the font type with which the Basic Script code is to be displayed within the edit window.
This item is available only after the Basic Script editor is opened.

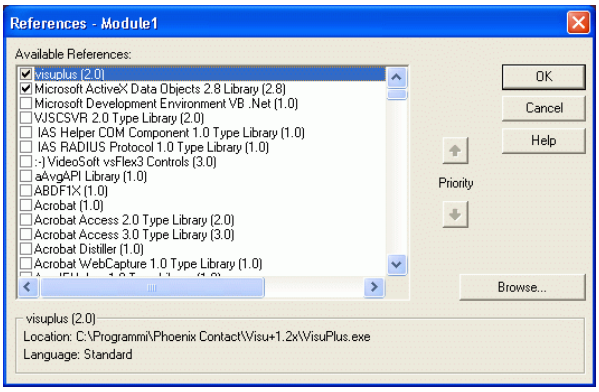
Syntax Colors

Allows you to change the standard colors of the Basic Script syntaxes which correspond to:

- **Light Turquoise:** Integrated VBA language functions
- **Green:** Comment lines
- **Red:** Instruction errors
- **Dark Blue:** Function added to Movicon or extended by using libraries external ActiveX/OCX
- **Blue:** Always reserved for integrated VBA engine functions

References...

Permits you to access the list of references (ActiveX/OCX) installed on the PC where the Movicon system is loaded. In addition to displaying this window you can also activate one or more of the references to implement their use in the basic script code.



Complete Word (CTRL+Space bar).

Displays a list of commands contained in the part of the text being edited. The shortcut command for this function is Ctrl+Space bar. When the space bar is pressed the command selected on the list is shown in the editing area.

Parameter Info (CTRL+SHIFT+Space Bar)

By positioning the cursor or a function and activating the command a tooltip is displayed showing the type of parameters needed for this function. This function is always active in the editing phase as well.

Open Uses

This command consents the functions and objects declared in a script's "#Uses" modules to be made available to the intellisense feature: (with command: CTRL + Space).

2.5.2. Debug Menu

The "Debug Menu" commands exclusively concern the Basic Script routine executions and are commands that normally found in the **"Basic Script ToolBar"** as well.

Step Into

This command executes the step into one function at a time each time the command is pressed.
This command is also available from the Movicon **"Basic Script ToolBar"**.

Step Over

This command executes steps over one function at a time when the command is pressed.
This command is also available from the Movicon **"Basic Script ToolBar"**.

Step Out

This command executes the presented functions until arriving at the line where the cursor is situated.
This command is also available in the Movicon **"Basic Script ToolBar"**.

Step To Cursor

This command executed the presented functions until arriving at the line where the cursor is situated.

Toggle Break

This command inserts or deletes a break point on line where the cursor is situated. This function is also available with a click on the furthest left border of the editor window corresponding to the line where the break point is to be inserted. The moment the basic executes a code line containing a break point, the execution will automatically switch into pause mode.
This command is also available in the Movicon **"Basic Script ToolBar"**.

Clear all Break

This command deletes all the Break Points inserted in the code.

Quick Watch

This command returns the result of the function which has been highlighted to a dialog window or the basic's debug window (Watch window).
This command is also available in the Movicon **"Basic Script ToolBar"**.

Add Watch

This command returns the result of the function which has been highlighted to a dialog window or the basic's debug window (Immediate Window).

Browse

This command opens the Movicon basic script function browse window.

Set Next Statement

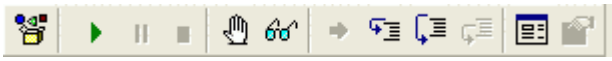
This command allows you to set the next instruction to be executed during the routine's debug phase. When activating this command it will be set as an instruction to execute the line where the mouse cursor is situated at that moment.

Show Next Statement

This command allows the cursor to be positioned at the top of the next instruction to be executed during the routine's debug phase.

2.5.3. Basic Script ToolBar

When opening the basic script code edit window the Basic Script ToolBar is shown at the top.



Some commands can also be executed from the "Debug Menu".
The keys' functions have been listed below:

Browse

This command is used for accessing the list of functions and properties available in the basic script. The window which appears is called Browse function.

Start/Resume

This command runs the basic script. When the code is referred to a project's basic resource the contents are run with the Sub Main otherwise if an object is referred to, the object's events management will be enabled.

Pause

This command pauses the basic script by positioning the cursor on the line being executed which will automatically be highlighted in yellow.

End

This command aborts the basic script run. The code must be paused beforehand in order for this command to work.

Break Point

This command inserts or deletes a break point on the line where the cursor is situated. This function is also available with a click on the extreme left border of the editor window corresponding to the line in which you want to insert a break point. The moment in which the basic script must execute a line of code with a break point it will be automatically put in pause.

Quick Watch

This command returns the result of the highlighted function, in a basic script dialog window or in the debug window (Watch window).

Show Current Statement

This command allows you to position the cursor at the top of the next instruction to be executed in the routine debug phase.

Step Into

This command executes the step into one function at a time, each time it is pressed.

Step Over

This command steps over one function at a time, each time it is pressed.

Step Out

This command executes the all the functions up to the line in which the cursor is situated.

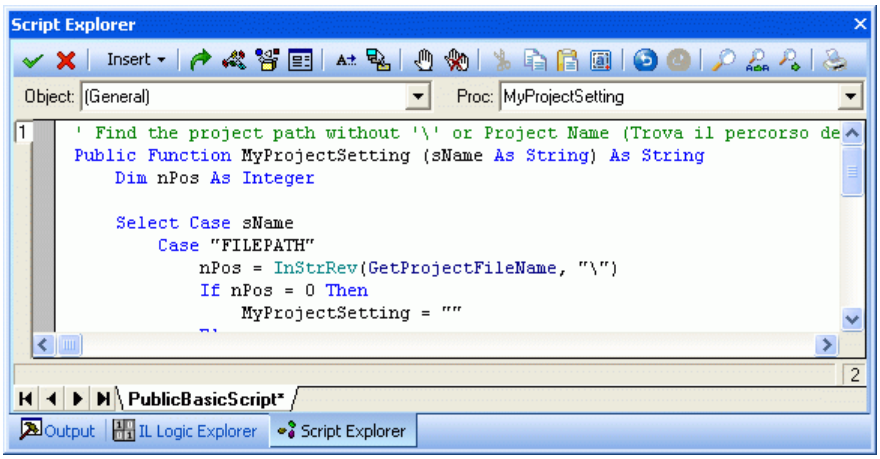
Edit UserDialog

This command opens the tool for creating the basic script's dialog window graphics. When exiting with OK, all that has been created graphically will be translated into code. When positioning and activating the command on this window all the new graphics will be reproduced from the translated basic script code.

Edit Module Properties

2.6. The Script Explorer Window

The "Script Explorer" window is fundamentally important for editing codes of the project's objects. This window in fact acts as a text editor to allow you to insert script codes for drawings, screens, alarms and also the Basic Script resources. The window's contents change dynamically according to the project's object or resource currently selected. You can associate the script code either to the project or to any single Movicon symbol or drawing. Therefore, by selecting a symbol its relating code, if inserted with one, will display in the "Script Explorer" window.



In addition to this, the F8 key has also been provided for calling the "Tag Browser" window, which is very handy to have when inserting variables.

By using the command bar at the tope of the "Script Explorer" window you can execute the code editing and debug commands:

The icons shown on the command bar mean:

Apply. Any changes made to the code will be made active by pressing this OK button.

Cancel. By pressing this button will delete any changes made to the code. Only the changes made after the last OK command execution will be cancelled.

Insert. This command allows RealTimeDB variable from the project to be inserted. The variable is selected from variable browse window which opens when pressing this button.

Run/Syntax. By pressing this button a check will be carried out on the inserted code and any errors found will be signalled (i.e syntax errors).

References. When this button is pressed the "Reference" window will open to select the ActiveX/OCX components to be inserted.

Browse. This command gives you access to the list of functions and properties available in the basic script by opening the Browse window.

Basic Script Dialog. This command opens the tool used for creating the basic script dialog window's graphics. When exiting with a OK to confirm, all the graphics created in codes will be entered. These codes will be reproduced graphically activating this command.

Complete Word. This command displays the list of commands available whose names begin with the same letters of the text being edited. The Ctrl+Space command can also be used. The command selected from the list is placed in the editing area with the Space key.

Parameter Info. By positioning the cursor on a function and activating the command a tooltip appears showing the type of parameters needed for that function. This function is also active in the editing phase.

Break Point. This command inserts or deletes a breakpoint in the line where the cursor is. This function is also available by clicking the far left border of the editor window in relation to the line in which the break point is to be inserted. When the basic has to execute a code line with a break point, the execution is automatically put into pause.

Clear all Break Point. This command deletes all the Break Points inserted into the code.

Open #Uses. This command allows functions and objects declared in a script's "#Uses" modules available to the intellisense (with command: CTRL + Space).

Cut. Cuts the text selected. In this case the text is cancelled but kept in memory on the Windows Clipboard.

Copy. Copies the text selected in memory on the Windows Clipboard.

Paste . Pastes the text previously Copied or Cut to the Windows Clipboard in the zone selected by the mouse in the workspace.

Select All. Executes the selection of all the texts contained in the "Script Explorer" window.

Undo. Undoes the last operation carried out (ie. deleting a text, etc).

Redo. Redoes the last operation cancelled (ie. cancelling a text, etc.).

Find. The Find command allows you to specify a character of text string to be searched for within the "Script Explorer" window.
You can specify whether the text to be searched for respects lower/Uppercase characters according to the standards.

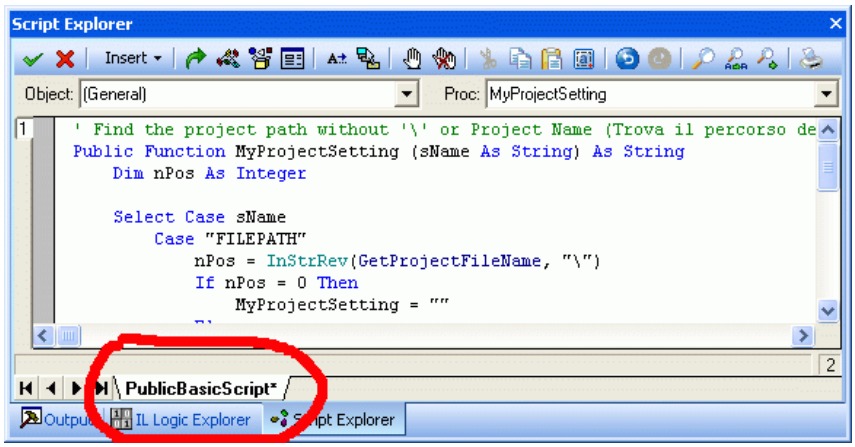
Replace. The Replace command lets you specify the test to be searched for and a text to replace it with.

Find Next. After having started a search with the Find command, you can search for the next specified text with the Find Next command.

Print. This command is used for printing the displayed code. A window will open for you to select and set the printer.

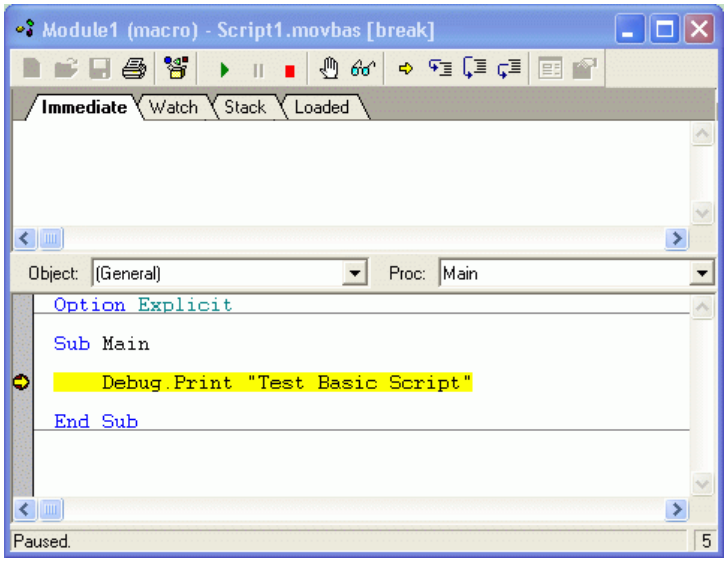
Object being Edited

When selecting a script, its name or the name of the associated component is shown in the bar at the bottom of the window. This let's you know which component is being referred to by the script displayed in the window:



2.7. Basic Script Debug

Movicon allows the complete debugging of the project's basic script logic, whether being project resources or contents in screen or alarm graphic symbols.
In the project design time phase you can debug the Basic Scripts set as Resources only, but you cannot test the scripts associated to objects or alarms as these are managed on event. However all these scripts can be debugged during the RunTime phase when Break-Points are inserted into their codes during the programming phase. Therefore when a script is executed in RunTime, a debug window will open when the execution stops at a Break-Point so that the script can be executed step by step, etc, by using the debug bar at the top.



Break-Points become permanent when inserted inside scripts, meaning that they are saved in the script's properties, and therefore they are also active in RunTime mode. So for this reason we advise you to insert Break-Points for carrying out the debug phase only and remove them straight after the test has been executed.

Some basic script functions verified in programming mode may return with different values when executed in runtime, therefore we advise you to use the debug in the programming phase to check codes roughly and repeat the test by executing the project in runtime to get an accurate test.

2.8. Basic Script Properties

Each **"Basic Script"** routine inserted as a resource in the "Project Explorer" window can receive in association Properties to setup how they function during a run. In order to this just select the "Basic Script" desired and then change its settings through the Movicon **"Properties Window"**.

2.8.1. Basic Script General Properties

By using the General properties you can set the name of the Basic Script resource selected in the "Project Explorer" window. In order to do this just select the Basic Script desired and then change its settings through the Movicon **"Properties Window"**.

Name

By using this property you can read or change the name of the Basic Script.

2.8.2. Basic Script Mode Properties

By using the "Mode" properties you can set the modalities with which the script will be executed.
To change the Mode properties of a Basic Script, select the object with the mouse and use the Movicon "**Properties Window**".

Run at Server

This property is to be used when the Redundancy between two projects is used. When enabled the script will be executed only in the project which has control at that moment. If, for instance, both the Primary and Secondary projects are being run, the Primary project will be in control. In this case, even though the execution of the Basic Script is commanded from the Secondary project it will be executed in the Primary project.
This management also happens when the script is executed from a "Child Project" set to connect to a "Network Server". Also in this situation the script is executed in the "Network Server" project instead of the Child Project.

UI Interface

This setting allows you to add the user interface management to the basic script. You need to enable this property when using controls such as "MsgBox" or "Dialog Box" inside the Basic Script.

The below listed WinWrap functions are not supported when the script's "Use User interface" property is not activated and cause an error when the AppActivate ,AboutWinWrapBasic,Shell,ShowPopupMenu script code is being loaded.

Separate Thread

This selection allows you to execute the Basin routine in a separate thread, therefore also at the same time as another basic routine or process on the same thread (dialog window).
This option is used when the basic script is run while another is already being run. If this option is not checked in both these basic scripts, the running of the second one will be buffered and run when the first one has terminated.
Please be reminded that not all the Movicon functions are available when run in a separate thread. Therefore we advise you to refer to the guide for each instruction to verify whether it can be executed or not in a separate thread.

Great care needs to be taken to the Scripts which do not use separate threads but have long operations to carry out. These Scripts are put in the same thread which they share (limited resource consumption), where they are, however, queued up in line for execution. Therefore you need to check the "Separate Thread" option for Scripts which execute long operations.

Create its Trace Tab

When this option box is enabled, a TAB will be created in the custom Output window, for the Basic Script during runtime mode, where messages inherent to the script are printed ie. the Debug.Print.

Modal Dialogs

When this option is enabled, the Dialog windows opened by the script can be made modal. Otherwise the dialog windows will not be modal and will disappear into the background when clicked outside.

2.8.3. Basic Script Execution Properties

By using the "Execution" properties you can setup the parameters with which the script will be run. To change the execution properties of a Script, first select the object with the mouse and then use the Movicon **"Properties Window**".

Priority

The execution of a basic script resource can be done with different priorities: Above Normal, Normal and Below normal:

Above Normal: Top priority
Normal: Normal Priority
Below Normal: Default value

Status Variable

You can assign one of the Movicon Real Time DB variables to the Basic Script Routine in which the system will write the routine's execution status as status value code in the variable desired.

The associated variable can be declared as any type (byte, word, etc.). The contents will be:

Bit 0 = Basic in run
Bit 1 = Basic in pause
Bit 2 = Basic in error
Bit 3 = Basic buffered
Bit 4 = Execution pending
Bit 5 = Basic stopped

The other bits are not handled, and are best left unused for any future handling. Note that only information on the Run status will be supplied when the variable is set as Bit type.

Maximum Instances

This property allows you to set the highest number of concurrent instances permitted for the Basic Script. This means that the Basic routine can also be called more than once at the same time and created with more instances. The final results will however be different according to how the "Separate Thread" property is set:

Separate Thread Property Enabled: if the script is executed many times at the same time, many concurrent threads will be created (up to the maximum amount defined in the "Max. Instances' property) and executed at the same time.
Separate Thread Property Disabled: if the script is executed many times at the same time, the requests will be buffered and executed immediately after the script has stopped being executed. If the script has not been set in a Separate Thread, only one instance at a time can be executed.

If afterwards a request is made to execute a script already in execution with a set maximum number of instances, this request will not be granted and notification will be given to historical log with this message:

"Cannot execute the script 'Basic Script1'. The script is already in execution or the maximum number of instances has been reached"

Sleep (ms)

The Sleep time is needed to lighten the workload of the basic script run on the processor. The longer the sleep time means lesser the workload from the basic script on the processor by slowing down.

Syncro Timeout (ms)

Timeout time for basic scripts executed in Syncro. This will be the maximum time waited before a timeout error is raised during script execution.

If the script resource is executed as startup basic script, it will get executed in syncro in respect to the other project resources started up as well (even with Separate Thread active), wating a maximum time equal to the one set in the parameter.

2.8.4. Script Debug Output

By using the "Script Debug Output" properties you can write on log files or print texts inserted in the Basic Script's "Debug.Print" instruction.
To change the Script's Script Debug Output, select the object with the mouse and use the Movicon **"Properties Window"**.

Status Bar

When enabling this property the messages executed by the Debug.Print function will also be printed in the Movicon Status Bar. This will also be done in the script's debug TAB if the **"Create its Trace Tab"** property has been enabled.

App.Log

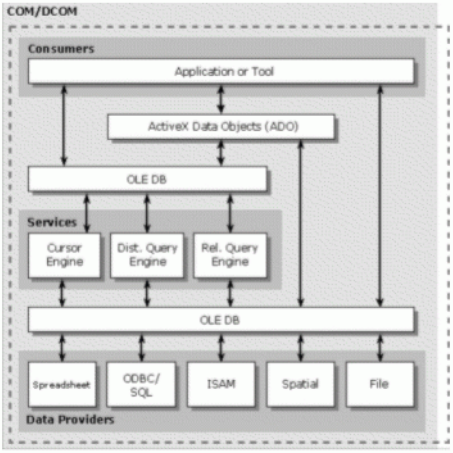
When enabling this property the messages executed by the Debug.Print function will also be printed in the project's Log.

Spooler

When enabling this property the messages executed by the Debug.Print function will be sent to the system's print spooler.

2.9. ADO in Basic Scripts

A Movicon Basic resource can make free use of the Microsoft ADO. objects (ActiveX Data Object) for direct access to data contained in a database.
The enabling of the ADO. functions can be done through the references by selecting the "Microsoft ActiveX Data Object Library..." item.
The ADO. functions permit interfacing with OLE DB systems which deal with retrieving information from databases. The Databases can be managed transparently, just as any object, internal a Basic Script.
The general architecture of integrating ADOs. into a system is as illustrated below:



A small example of a basic script code which uses A:DO. for accessing a DataBase is given below:

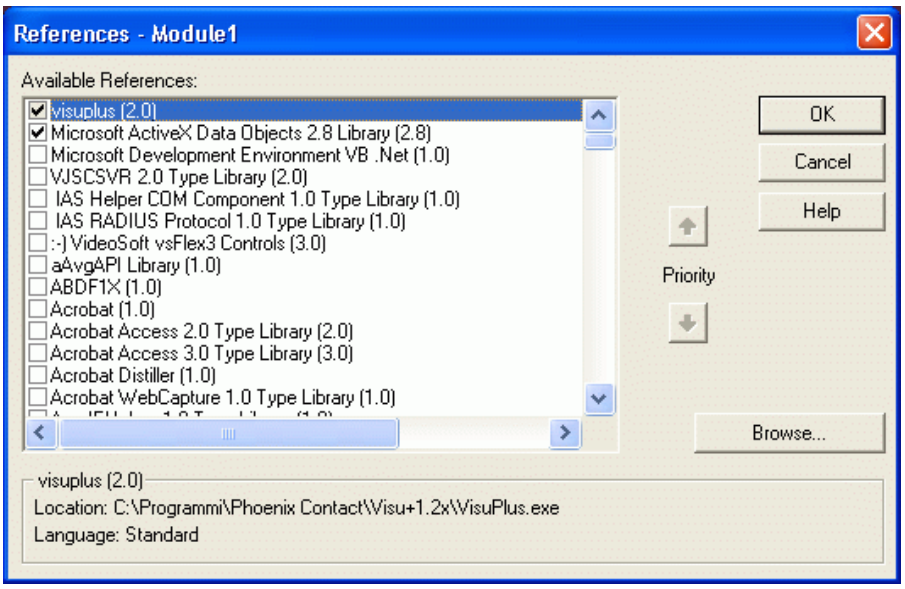
```
Option Explicit
Const PATH_DB As String = "\DataAdo\"
Const FILE_DB As String = "DataAdo.mdb"

'Data for ADO. architecture
Dim Conn1 As New ADODB.Connection
Dim Rs1 As New ADODB.Recordset

Sub Main
        Dim sAccessConnect As String
        Dim lNumCariche As Long
        Dim sQuery As String

        'ODBC connection parameters
        sAccessConnect = "Provider=Microsoft.Jet.OLEDB.4.0;Data " & _
                                "Source=" & CurDir & PATH_DB & FILE_DB & _
                                ";User ID=Admin;Password=;"

        'Open Connection
        Conn1.ConnectionString = sAccessConnect
        Conn1.Open

        sQuery = "SELECT Cariche.* " & _
                        "FROM Cariche " & _
                        "WHERE ID=1;"

        'Recordset Creation
        Rs1.CursorType = adOpenKeyset
        Rs1.LockType = adLockOptimistic
        Rs1.Open sQuery, Conn1, , , adCmdText

        If Not Rs1.EOF Then
                SetVariableValue("VAR00001",Rs1.Fields("RecipeAT").Value)
        End If

        Rs1.Close
        Set Rs1 = Nothing
        Conn1.Close
        Set Conn1 = Nothing
End Sub
```

In this example you can see how to access data, contained in a MS Access table, by means of using the ADO. functionalities.
The "DataAdo.mdb" file contains the "Cariche" table which contains the "RecipeAT" field.
By using the SQL syntax we can select the "cariche" (load) record with ID=1 and extract the relative AT recipe code.
The variables needed are "Conn1" Connection type", designated to pointing the file by using the "Open" method and the "Rs1" Recordset type variable designated to contain the record or record set which satisfies the SQL query selection. The "Conn1" and "Rs1" variables are destroyed at the end of the routine by using the "Close" and "Nothing" methods respectively to leave memory allocated for the next time it is created.
The ADO. engine is thread safe and can be used in more threads at the same time.

2.10. ActiveX/OCX in Basic Scripts

The ActiveX/OCX objects can be used inside Basic Script routines. A Movicon Basic resource can freely use the ActiveX/OCX objects, objects realized by third parties and independent of Movicon for creating more varied functions.
The advantages offered by using the Microsoft ActiveX/OCX technology, such as in ADO, are enormous and indisputable, allowing the programmer to reuse their codes in different Container applications.

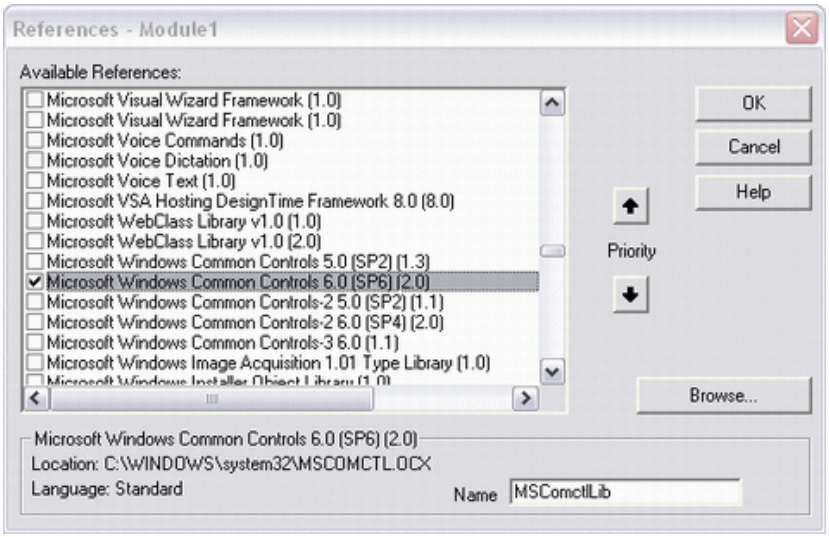
Before managing an external object within a Basic Script, a link must always be created, by using the CreateObject function, or even better by using the References links. In the latter, the object's properties are available directly from within the Browser of the Movicon functions library.
To execute the link, select the References item from the Movicon Edit menu when the Basic resource is active. The window, shown below, will open through which the link to the desired object can be checked.



2.10.1. How to use ActiveX with Licenses

Some ActiveX objects need a license in order to work. In this case the "CreateObjectLic()" script function is used for creating an ActiveX object with a license. This function requires two parameters, the ActiveX Server name and its license. If these two parameters are unknown, they can be retrieved in the following way:

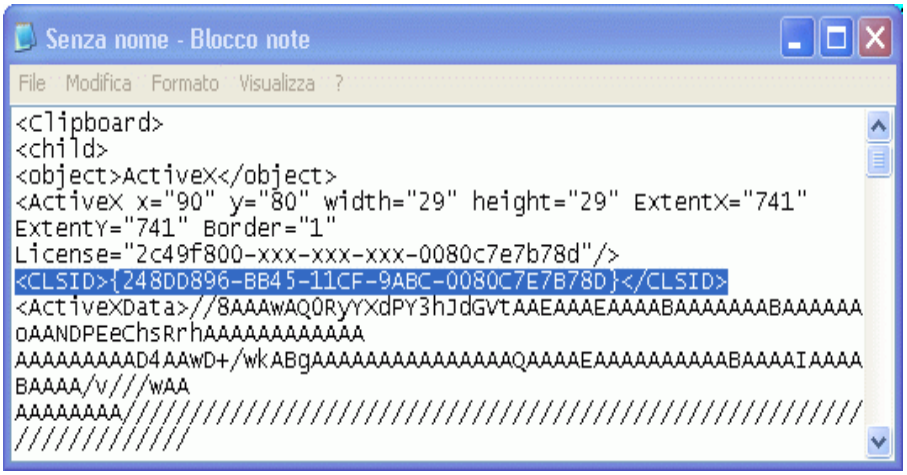
1. Go to Basic Script and Enable the Reference of the ActiveX to be used. For instance, Microsoft WinSock:



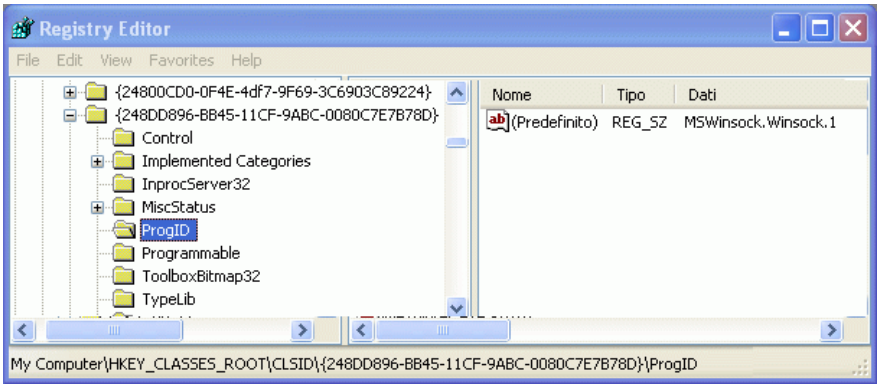
2. Temporarily insert an ActiveX on screen by going to the Toolbox "Advance Shapes" group, click the ActiveX object to open the "Insert ActiveX Object" window and select and insert the one desired, ie. Microsoft WinSock. Go to its properties window and click the "Get ActiveX

license" to view the license code. This code is the one to be inserted in the "CreateObjectLic()" function's "lpszLicense" parameter.

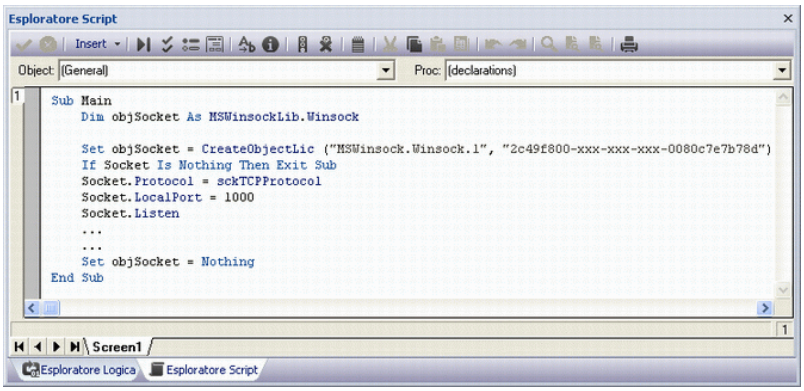
3. To retrieve th ActiveX Server name, copy the Active object you inserted on screen to any text editor such as Notepad. Then search for the object's CLSID:



4. Now open the Windows registry and look for the key corresponding to the CLSID and then the "ProgID" subkey. The ActiveX Server name is shown in the predefined value which is to be inserted in the "CreateObjectLic()" function's "lpszServerName":



5. Cancel the ActiveX object from the screen and insert the required code in the Basic Script:



2.10.2. Example of using an ActiveX/OCX into a Basic Script

The following example shows how to use the Microsoft object "MSCOMM32.OCX" to manage the serial port within a Basic Script resource.
Remember that Movicon allows to manage the serial port using the internal VBA functions "IOPortInterface".
This is only an example.

```
Option Explicit

Const BASESTX As Byte = &H20
Const COM_SCRIVI As String = "Q"
Const TIMEOUT As Long = 3 'secondi
Const MAX_TENTATIVI As Long = 5

Sub Main
	'##################################
	'CREATING MESSAGE STRING
	'##################################

	Dim message As String
	Dim dati As Long
	Dim address As Byte
	Dim numByte As Byte
	Dim chk As Long
	Dim i As Integer

	Begin Dialog UserDialog 270,105,"INSERT PARAMETERS" ' %GRID:10,5,1,1
	Text 10,10,90,20,"Address",.Add
	TextBox 140,10,110,20,.address
	Text 10,40,90,20,"Dato",.dato
	TextBox 140,40,110,20,.dati
	OKButton 30,70,90,25
	CancelButton 150,70,90,25
	End Dialog
	Dim dlg As UserDialog

	If Dialog (dlg) = 0 Then Exit Sub
	If IsNumeric(CVar(dlg.dati)) And IsNumeric(CVar(dlg.address)) Then
		address = CByte (dlg.address)
		dati = CLng (dlg.dati)
	Else
		MsgBox ("PARAMETERS NOT VALID", vbCritical, "ERROR")
		Exit Sub
	End If

	'calculation data byte nr. as lenght of the string hex, dividing as integer by 4
	'(char. Hex into a word) approsimation by exceed (+3) all of this multiplicated by 2 (byte for
	word)
	numByte = ((Len(Hex(dati))+3)\4)*2
	message = Chr(BASESTX + numByte) & COM_SCRIVI
	message = message & Format(address,"00")
	message = message & String((numByte*2-Len(Hex(dati))), "0") & Hex(dati)

	chk = 0
	For i = 2 To Len(message) 'exclude the first character (STX)
		chk = chk + Asc(Mid(message,i,1))
	Next i

	message = message & Right(Hex(chk),2) & vbCr

	'##################################
	'SEND TO SERIAL PORT
	'##################################
	' oggetto per l'OCX MSComm32
	Dim MSComm1 As Object
	' buffer for the input string
```

```
        Dim InString As String
        ' time of message sending
        Dim oraInvio As Date
        ' number of trial to send
        Dim numTentativi As Long

         Set MSComm1 = CreateObject("MSCOMMLib.MSComm.1")

         ' Use COM2.
         MSComm1.CommPort = 2
        ' 9600 baud, no parity, 8 data, and 1 stop bit.
        MSComm1.Settings = "9600,N,8,1"
        ' opening serial port
        MSComm1.PortOpen = True
        ' preparing to read the complete input buffer
        MSComm1.InputLen = 0
        ' reset counter trial send with timeout
        numTentativi = 0
Invio:
        'reset buffer of input
        MSComm1.InBufferCount = 0
        ' send message string
        MSComm1.Output = message
        ' set the time of sending message
        oraInvio = Now

        ' waiting the return of all data to the serial port
        Do
                DoEvents
        Loop    Until    MSComm1.InBufferCount    =    Len(message)    Or    Now    >
        DateAdd("s",TIMEOUT,oraInvio)

        If MSComm1.InBufferCount <> 0 Then
                ' reading data form input buffer
                InString = MSComm1.Input
                If InString <> message Then
                        numTentativi = numTentativi + 1
                        If numTentativi < MAX_TENTATIVI Then
                                GoTo Invio
                        Else
                                MsgBox("Transmission Error",vbCritical,"ERROR")
                        End If
                End If
        Else
                MsgBox("Time Out communication",vbCritical,"ERROR")
        End If
        ' closing the serial port
        MSComm1.PortOpen = False
End Sub
```

3. API Basic Interface

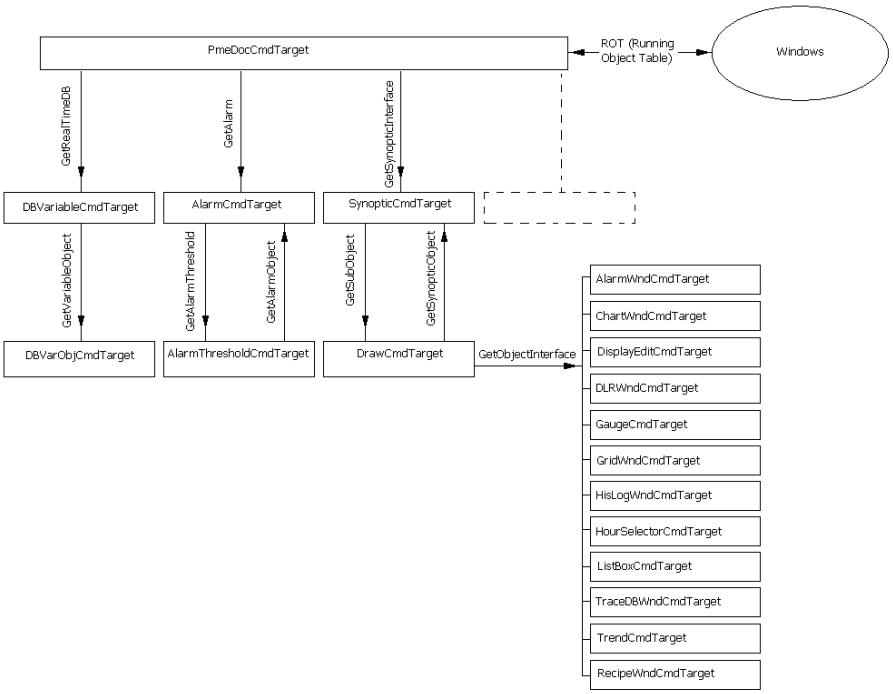
3.1. Using Basic Script Interfaces

Movicon consents the use of a series of basic script methods and properties for managing projects and the resources and objects they contain. In this way you can read/modify almost all the properties of the project resources and objects in runtime mode. Each project resource and object has been specifically provided with a set of methods and properties for this purpose and are found in the objects' Basic Script Interface group properties.

According to the basic script context you are working in, ie. editing code in a Basic Script resource, or in a screen object, etc, you will be provided with the methods and properties of the interface relative to that object. In order to have methods and properties from a different interface you will need to create a reference object to the new interface you wish to use. For instance, if you are editing script code in a screen, you will be provided with the "**SynopticCmdTarget**" interface. However, if you then need to edit a title of a symbol contained in that screen, you will have to create an "**DrawCmdTarget**" object using the "GetSubObject" function.

You must also take into account that the methods and properties of the main "PmeDocCmdTarget" interfaces (general project interfaces), "UIInterface" (project user interface) and "DBVariableCmdTarget" (RealTimeDB database of the project's variables) are always available independently from the context you are working in.

The "**PmeDocCmdTarget**" interface contains a series of methods which consent to project resource referencing (Alarms, RealTimeDB, Screens, DataLogger/Recipes, etc.). The flow chart below shows how to access objects and resources (only some have been used as examples but this procedure is the same for all the basic script interface) starting from the "**PmeDocCmdTarget**".



These Basic Script Interfaces are listed below with the object type they are referenced to.

Basic Script Interface	Description
AlarmCmdTarget	This interface groups the methods and properties of "Alarm" objects.
AlarmThresholdCmdTarget	This interface groups the events, methods and properties of "Alarm Threshold" objects.
AlarmWndCmdTarget	This interface groups the events, methods and properties of "Alarm Window" objects.
ChartWndCmdTarget	This interface groups the events, methods and properties of "Chart" objects.
ClientRulesInterface	This interface groups the events, methods and properties of Networking "Client Rules" objects.
CommandLanguageCmdTarget	This interface groups the methods and the properties which allow screen button or object command list "language" commands to be modified.
CommandsListCmdTarget	This interface groups the method and propertied that allow screen button or object command lists to be modified.
ButtonCmdTarget	This interface groups the methods or properties which allow Button object properties to be modified.
CommandUsersCmdTarget	This interface groups methods and properties that allow "Users" command types on Screen Button or Object Command Lists to be modified.
CommandAlarmCmdTarget	This interface groups methods and properties that allow "Alarm" command types on Screen Button or Object Command Lists to be modified.
CommandVariableCmdTarget	This interface groups methods and properties that allow "Variable" command types on Screen Button or Object Command Lists to be modified.
CommandBaseCmdTarget	This interface groups methods and properties common to all script interfaces usded for modifying individual commands from screen Button or Object command lists.
CommandHelpCmdTarget	This interface groups methods and properties that allow "Help" command types on Screen Button or Object Command Lists to be modified.
CommandMenuCmdTarget	This interface groups methods and properties that allow "Menu" command types on Screen Button or Object Command Lists to be modified.
CommandReportCmdTarget	This interface groups methods and properties that allow "Report/Recipe" command types on Screen Button or Object Command Lists to be modified.
CommandScriptCmdTarget	This interface groups methods and properties that allow "Script" command types on Screen Button or Object Command Lists to be modified.
CommandSynopticCmdTarget	This interface groups methods and properties that allow "Screen" command types on Screen Button or Object

	Command Lists to be modified.
CommandSystemCmdTarget	This interface groups methods and properties that allow "System" command types on Screen Button or Object Command Lists to be modified.
CommandEventCmdTarget	This interface groups methods and properties that allow "Event" command types on Screen Button or Object Command Lists to be modified.
DBVariableCmdTarget	This interface groups the methods and properties for accessing the RealTimeDB settings.
DBVarObjCmdTarget	This interface groups the methods and properties of RealTimeDB "Tag" objects.
DisplayEditCmdTarget	This interface groups events, methods and properties of "Display" objects. You must take in to consideration that a "List Box" or "Combo Box" object inherits methods and properties both from the "DisplayEditCmdTarget" and "ListBoxCmdTarget" interfaces.
DLRCmdTarget	This interface groups methods and properties of "DataLogger" or "Recipe" objects.
DLRColumnCmdTarget	This interface groups methods and properties of DataLogger or Recipe "Column" objects.
DLRWndCmdTarget	This interface groups events, methods and properties of "DataLogger-Recipe Window" objects.
DrawCmdTarget	This interface groups generic events, methods and general properties for all objects that can be inserted on screen.
EventCmdTarget	This interface groups methods, properties and project resource "Event" objects.
GaugeCmdTarget	This interface groups events, methods and properties of "Gauge" objects.
GenericEvents	This group of events are generic and can be partly or completely available for all those objects which can be inserted on screen.
GridWndCmdTarget	This interface groups "Grid" object events, methods and properties.
HisLogWndCmdTarget	This interface groups events, methods and properties of "Historical Log Window" objects.
HourSelectorCmdTarget	This interface groups events, methods and properties of "Scheduler Window" objects.
IOPortInterface	This interface groups methods and properties which consent access to PC COM serial ports.
ListBoxCmdTarget	This interface groups events, methods and properties of "List Box" or "Combo Box" objects.
NetworkClientCmd	This interface groups methods and properties which consent access to project's "Network Client" settings.

NetworkRedudancyCmd	This interface groups methods and properties which consent access to the project's "Redundancy" settings.
OPCAECmdTarget	Momentarily not supported.
OPCClientCmdTarget	This interface groups methods and properties which consent access to the project's OPC client general settings.
OPCClientGroupObjCmdTarget	This interface groups methods and properties which consent access to the project's OPC Client "Group" settings.
OPCClientItemObjCmdTarget	This interface groups methods and properties which consent access to the project's OPC Client "Item" settings.
OPCClientObjCmdTarget	This interface groups methods and properties which consent access to a specific project OPC Client settings.
OPCServerCmdTarget	This interface groups methods and properties which consent access to the Movicon OPC Server settings.
PmeDocCmdTarget	This interface groups the project's generic methods and properties.
RASStationInterface	This interface groups Networking "RAS Station" methods and properties.
RecipeWndCmdTarget	This interface regroups the "Recipe Window Manager" object's events, methods and properties.
ScalingCmdTarget	This interface groups methods and properties of the project's resources' "Scaling" objects.
SchedulerCmdTarget	This interface groups the methods and properties of the project's resources' "Scheduler" objects.
ScriptMEIterface	This interface groups the events, methods and properties of the project's "Basic Script" resource.
SynopticCmdTarget	This interface groups events, methods and properties of the project's "Screen" resource.
TraceDBWndCmdTarget	This interface groups events, methods and properties of "TraceDB Window" objects.
TrendCmdTarget	This interface groups events, methods and properties of "Trend" or "Data Analysis" objects.
UIInterface	This interface groups the project's generic methods and properties concerning the user interface.
UserAndGroupCmdTarget	This interface groups the project's "User Management" generic methods and properties.
UserCmdTarget	This interface groups the project's "User" object methods and properties.
UserGroupCmdTarget	This interface groups the project's "User Group" object methods and properties.

4. AlarmCmdTarget

GetAlarmThreshold, AlarmCmdTarget Function

Syntax GetAlarmThreshold(_lpszName)

Description Returns an AlarmThresholdCmdTarget object type for the alarm threshold management.

Parameter	Description
lpszName As String	Name of threshold to be retrieved.

Result Object
If Function has been executed successfully it will retrieve an object of type AlarmThresholdCmdTarget if otherwise Nothing is returned.

Example:

```
Public Sub AlarmLoading()
        Dim objAlarm As AlarmCmdTarget
        Dim objThreshold As AlarmThresholdCmdTarget
        Set objAlarm = GetAlarmObject
        Set objThreshold = objAlarm.GetAlarmThreshold("High")
        Debug.Print objThreshold.BackColor
        Set objThreshold = Nothing
        Set objAlarm = Nothing
End Sub
```

GetXMLSettings, AlarmCmdTarget Function

Syntax GetXMLSettings()

Description This function returns the Alarm object's string definition in XML format.

Parameter	Description
None	None

Result String

Example:

```
Public Sub AlarmLoading()
        Dim objAlarm As AlarmCmdTarget
        Dim sResult As String
        Set objAlarm = GetAlarmObject
```

```
        sResult = objAlarm.GetXMLSettings
        MsgBox("XMLSetting: " & sResult, vbOkOnly, GetProjectTitle)
        Set objAlarm = Nothing
End Sub
```

AlarmOnQualityGood, AlarmCmdTarget Property

Syntax AlarmOnQualityGood = _Boolean

Description Enables or disables the alarm management only when the quality of the variable linked to it is good. This means, for example, that the alarms associated to variables which arrive directly from the PLC or from any other field device, will be automatically disabled when communication is interrupted.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub AlarmLoading()
        Dim objAlarm As AlarmCmdTarget
        Dim bResult As Boolean
        Set objAlarm = GetAlarmObject
        bResult = objAlarm.AlarmOnQualityGood
        Debug.Print bResult
        Set objAlarm = Nothing
End Sub
```

DeviceName, AlarmCmdTarget Property

Syntax DeviceName

Description This property sets or returns the name of the device associated to the alarm. This property is only in read.

Parameter	Description
None	None

Result String

Example:

```
Public Sub AlarmLoading()
        Dim objAlarm As AlarmCmdTarget
        Dim sResult As String
        Set objAlarm = GetAlarmObject
        sResult = objAlarm.DeviceName
        Debug.Print sResult
```

```
        Set objAlarm = Nothing
End Sub
```

Enabled, AlarmCmdTarget Property

Syntax Enabled = _Boolean

Description Enables or disables alarm.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub AlarmLoading()
        Dim objAlarm As AlarmCmdTarget
        Dim bResult As Boolean
        Set objAlarm = GetAlarmObject
        bResult = objAlarm.Enabled
        Debug.Print bResult
        Set objAlarm = Nothing
End Sub
```

EnableVariableName, AlarmCmdTarget Property

Syntax EnableVariableName

Description This property returns, in string format, the name of the enabling variable associated to the alarm. This property is in read only.

Parameter	Description
None	None

Result String

Example:

```
Public Sub AlarmLoading()
        Dim objAlarm As AlarmCmdTarget
        Dim sResult As String
        Set objAlarm = GetAlarmObject
        sResult = objAlarm.EnableVariableName
        Debug.Print sResult
        Set objAlarm = Nothing
End Sub
```

Isteresis, AlarmCmdTarget Property

Syntax Isteresis = _Long

Description This property returns the alarm's isteresis value. This property is in read only.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub AlarmLoading()
        Dim objAlarm As AlarmCmdTarget
        Dim lResult As Long
        Set objAlarm = GetAlarmObject
        lResult = objAlarm.Isteresis
        Debug.Print sResult
        Set objAlarm = Nothing
End Sub
```

Name, AlarmCmdTarget Property

Syntax Name

Description This property sets or returns the name of the alarm.

Parameter	Description
None	None

Result String

Example:

```
Public Sub AlarmLoading()
        Dim objAlarm As AlarmCmdTarget
        Dim sResult As String
        Set objAlarm = GetAlarmObject
        sResult = objAlarm.Name
        Debug.Print sResult
        Set objAlarm = Nothing
End Sub
```

ThresholdExclusive, AlarmCmdTarget Property

Syntax ThresholdExclusive = _Boolean

Description When this property is enabled the alarm referred to the threshold is silenced when the alarm referred to the next threshold occurs. Otherwise, both alarms will be kept active.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub AlarmLoading()
        Dim objAlarm As AlarmCmdTarget
        Dim bResult As Boolean
        Set objAlarm = GetAlarmObject
        bResult = objAlarm.ThresholdExclusive
        Debug.Print bResult
        Set objAlarm = Nothing
End Sub
```

VariableName, AlarmCmdTarget Property

Syntax VariableName

Description This property returns, in string format, the name of the variable associated to the alarm. This property is in read only.

Parameter	Description
None	None

Result String

Example:

```
Public Sub AlarmLoading()
        Dim objAlarm As AlarmCmdTarget
        Dim sResult As String
        Set objAlarm = GetAlarmObject
        sResult = objAlarm.VariableName
        Debug.Print sResult
        Set objAlarm = Nothing
End Sub
```

5. AlarmThresholdCmdTarget

AlarmLoading, AlarmThresholdCmdTarget Event

Description Event occurs when the alarm is initialized due to the project going into run mode.

Parameter	Description
None	None

AlarmUnloading, AlarmThresholdCmdTarget Event

Description Event occurs when the alarm object is unloaded from memory because the project has stopped running.

Parameter	Description
None	None

OnAckAlarm, AlarmThresholdCmdTarget Event

Description Event occurs when the alarm is acknowledged with its Ack command. The bRet boolean variable is managed in the event to allow or disallow the execution of the alarm acknowledgement, ie. when the bRet is set at False the alarm will not be acknowledged.

Parameter	Description
bRet As Boolean	Enables Alarm acknowledgement.

OnCommentAlarm, AlarmThresholdCmdTarget Event

Description Event occurs when the comment associated to the alarm/message is changed or added to. This operation can be done in the alarms or messages window by using the appropriate commands.

The bRet variable allows this event to be locked out, preventing the comment to be added to or edited.

Parameter	Description
bRet As Boolean	Enables comment editing

OnHelpAlarm, AlarmThresholdCmdTarget Event

Description Event occurs when a request to view the help string is made on the alarm/message object. This operation can be done from the alarms or messages window by using the appropriate commands.
The bRet variable allows this event to be locked out preventing the help text to be displayed.

Parameter	Description
bRet As Boolean	Enables the help text display

OnResetAlarm, AlarmThresholdCmdTarget Event

Description Event occurs when the alarm's reset command is executed. The bRet boolean variable allows or disallows alarm reset execution, ie. when the bRet is set at False the alarm will not be reset.

Parameter	Description
bRet As Boolean	Enables alarm reset

OnSetAlarm, AlarmThresholdCmdTarget Event

Description Event occurs when alarms goes on or off. The bSet boolean variable returns the alarm status (ON = True, OFF = False) while the bRet boolean variable allows or disallows the alarm to go on or off, ie. when the bRet is set at False is will not be able to go ON, and viceversa when set at OFF.

Parameter	Description
bSet As Boolean	Alarm status
bRet As Boolean	Enables the alarm's status change

ResetAlarm, AlarmThresholdCmdTarget Function

Syntax ResetAlarm()

Description Resets the alarm relating to the reference threshold.

Parameter	Description
None	None

Result None

Example:

```
Public Sub OnAckAlarm()
        '..
        ResetAlarm
        '..
End Sub
```

GetAlarmObject, AlarmThresholdCmdTarget Function

Syntax GetAlarmObject()

Description Gets the alarm objet relating to the reference threshold.

Parameter	Description
Noneuno	Noneuno

Result Object
If Function has been executed successfully it will retrieve an object of type AlarmCmdTarget if otherwise Nothing is returned.

Example:

```
Public Sub AlarmLoading()
        Dim objAlarm As AlarmCmdTarget
        Dim sResult As String
        Set objAlarm = GetAlarmObject
        sResult = objAlarm.GetXMLSettings
        MsgBox("XMLSetting: " & sResult, vbOkOnly, GetProjectTitle)
        Set objAlarm = Nothing
End Sub
```

GetTotNumAck, AlarmThresholdCmdTarget Function

Syntax GetTotNumAck()

Description This property returns the total number of times the alarm has been acknowledged.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub OnAckAlarm()
        Dim objAlarm As AlarmCmdTarget
        Dim objAlarmThreshold As AlarmThresholdCmdTarget
        Set objAlarm = GetAlarm("Alarm01")
        Set objAlarmThreshold = objAlarm.GetAlarmThreshold("Threshold01")
        MsgBox " GetTotNumAck = " &
        CStr(objAlarmThreshold.GetTotNumAck),vbInformation,GetProjectTitle
        Set objAlarm = Nothing
        Set objAlarmThreshold = Nothing
End Sub
```

GetTotNumOn, AlarmThresholdCmdTarget Function

Syntax GetTotNumOn()

Description This property returns the total number of time the alarm turned ON.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub OnAckAlarm()
        Dim objAlarm As AlarmCmdTarget
        Dim objAlarmThreshold As AlarmThresholdCmdTarget
        Set objAlarm = GetAlarm("Alarm01")
        Set objAlarmThreshold = objAlarm.GetAlarmThreshold("Threshold01")
        MsgBox " GetTotNumOn = " &
        CStr(objAlarmThreshold.GetTotNumOn),vbInformation,GetProjectTitle
        Set objAlarm = Nothing
        Set objAlarmThreshold = Nothing
End Sub
```

GetTotNumReset, AlarmThresholdCmdTarget Function

Syntax GetTotNumReset()

Description This property returns the total number of times the alarm has been reset.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub OnAckAlarm()
        Dim objAlarm As AlarmCmdTarget
        Dim objAlarmThreshold As AlarmThresholdCmdTarget
        Set objAlarm = GetAlarm("Alarm01")
        Set objAlarmThreshold = objAlarm.GetAlarmThreshold("Threshold01")
        MsgBox " GetTotNumAck = " & CStr(objAlarmThreshold.GetTotNumReset),vbInformation,GetProjectTitle
        Set objAlarm = Nothing
        Set objAlarmThreshold = Nothing
End Sub
```

GetTransactionID, AlarmThresholdCmdTarget Function

Syntax GetTransactionID()

Description This property returns the Transaction ID number that the alarm has reached.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub OnAckAlarm()
        Dim objAlarm As AlarmCmdTarget
        Dim objAlarmThreshold As AlarmThresholdCmdTarget
        Set objAlarm = GetAlarm("Alarm01")
        Set objAlarmThreshold = objAlarm.GetAlarmThreshold("Threshold01")
```

```
        MsgBox " GetTransactionID = " &
        CStr(objAlarmThreshold.GetTransactionID),vbInformation,GetProjectTitle
        Set objAlarm = Nothing
        Set objAlarmThreshold = Nothing
End Sub
```

GetUniqueID, AlarmThresholdCmdTarget Function

Syntax GetUniqueID()

Description This property returns the unique ID number associated to the alarm.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub OnAckAlarm()
        Dim objAlarm As AlarmCmdTarget
        Dim objAlarmThreshold As AlarmThresholdCmdTarget
        Set objAlarm = GetAlarm("Alarm01")
        Set objAlarmThreshold = objAlarm.GetAlarmThreshold("Threshold01")
        MsgBox " GetUniqueID = " &
        CStr(objAlarmThreshold.GetUniqueID),vbInformation,GetProjectTitle
        Set objAlarm = Nothing
        Set objAlarmThreshold = Nothing
End Sub
```

GetXMLSettings, AlarmThresholdCmdTarget Function

Syntax GetXMLSettings()

Description This function returns the alarm object's definition string XML format.

Parameter	Description
None	None

Result String

Example:

```
Public Sub AlarmLoading()
        Dim sResult As String
```

```
        sResult = GetXMLSettings
        MsgBox("XMLSetting: " & sResult, vbOkOnly, GetProjectTitle)
End Sub
```

AckAlarm, AlarmThresholdCmdTarget Function

Syntax AckAlarm()

Description Executes the acknowledgement (Ack) of the alarm relating to the reference threshold.

Parameter	Description
None	None

Result None

Example:

```
Public Sub AlarmLoading()
        AckAlarm
End Sub
```

AlarmArea, AlarmThresholdCmdTarget Property

Syntax AlarmArea = _String

Description This property sets or returns the alarms area or messages area to which the string, associated to the threshold's text, belongs to. Accepts a string type parameter.

Parameter	Description
None	None

Result String

Example:

```
Public Sub AlarmLoading()
        Debug.Print AlarmArea
End Sub
```

Attachment, AlarmThresholdCmdTarget Property

Syntax Attachment = _String

Description The file (one or more than one), to be attached to the message to be sent to the recipient, can be retrieved or set through this property. When there are more than one file you must use the ";" character as separator (ie. File1.zip;File2.zip;File3.zip).

Parameter	Description
None	None

Result String

Example:

```
Public Sub AlarmLoading()
        Debug.Print Attachment
End Sub
```

BackColor, AlarmThresholdCmdTarget Property

Syntax BackColor = _Long

Description This property sets or returns the back color relating to the alarm or message displayed in the appropriate window.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub AlarmLoading()
        Debug.Print BackColor
End Sub
```

Beep, AlarmThresholdCmdTarget Property

Syntax Beep = _Boolean

Description This property allows the warning sound, which has been generated by the computer's buzzer when the alarm goes ON.

Parameter	Description
None	None

Result Boolean

Example:
```
Public Sub AlarmLoading()
        Debug.Print Beep
End Sub
```

BlinkBackColor, AlarmThresholdCmdTarget Property

Syntax BlinkBackColor = _Long

Description This property sets or returns the back color relating to the alarm or message displayed in the appropriated window during the blink phase.

Parameter	Description
None	None

Result Long

Example:
```
Public Sub AlarmLoading()
        Debug.Print BlinkBackColor
End Sub
```

BlinkOnNewAlarm, AlarmThresholdCmdTarget Property

Syntax BlinkOnNewAlarm = _Boolean

Description This property is used for activating or deactivating the alarm's blink function when it is activated.

Parameter	Description
None	None

Result Boolean

Example:
Public Sub AlarmLoading()
    Debug.Print **BlinkOnNewAlarm**
End Sub

BlinkTextColor, AlarmThresholdCmdTarget Property

Syntax BlinkTextColor = _Long

Description This property sets or returns the color to be associated to the text relating to the alarm or message displayed in the appropriate window during the blink phase.

Parameter	Description
None	None

Result Long

Example:
Public Sub AlarmLoading()
    Debug.Print **BlinkTextColor**
End Sub

CommandList..., AlarmThresholdCmdTarget Property

Syntax CommandList... = _String

Description This property returns the project's XML string containing the definition of the "Comand List" associated to the reference alarm threshold which should be executed on the respective event:

CommandList: the command list is executed on user request when the alarm is active. Only in this case, From the Alarm Window, by double clicking with the mouse on the alarm while pressing the CTRL key at the same time, will the Command list be executed.
CommandListAck: the command list is executed on the alarm's acknowledge event.
CommandListOff: the commmand list is executed on the alarm's deactivation (OFF) event.
CommandListOn: the command list is executed on the alarm's activation (ON) event.
CommandListReset: the command list is executed on the alarm's reset event.

Parameter	Description
None	None

Result String

Example:

```
Public Sub AlarmLoading()
        Debug.Print CommandList
        Debug.Print CommandListAck
        Debug.Print CommandListOff
        Debug.Print CommandListOn
        Debug.Print CommandListReset
End Sub
```

CommentOnAck, AlarmThresholdCmdTarget Property

Syntax CommentOnAck = _Boolean

Description Consents you to set or rest this option for inserting an alarm acknowledge comment.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        im objAlarmWnd As AlarmWndCmdTarget
        Dim objAlarmThr As AlarmThresholdCmdTarget
        Set objAlarmWnd =
        GetSynopticObject.GetSubObject("AlarmWnd").GetObjectInterface
        Set objAlarmThr = objAlarmWnd.GetSelectedAlarm
        If Not objAlarmThr Is Nothing Then
                objAlarmThr.CommentOnAck = Not objAlarmThr.CommentOnAck
        End If
End Sub
```

Condition, AlarmThresholdCmdTarget Property

Syntax Condition = _Integer

Description This proprety sets or returns the condition for verifying referenced alarm. The possible configurations are:

0 = major-equal (>=)
1 = minor-equal (<=)
2 = equal (=)
3 = Rate Change Decrease
4 = Rate Change Incease

5 = Different (<>)
6 = Between

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub AlarmLoading()
        Msgbox "Alarm Condition = " & cstr(Condition), vbInformation, GetProjectTitle
End Sub
```

DateTimeACK, AlarmThresholdCmdTarget Property

Syntax DateTimeACK = _Date

Description This property sets or returns the date and time in which the alarm acknowledgement took place. Accepts a date parameter type. This property changes the value displayed in the alarms window only and has not influence in the historical log.

Parameter	Description
None	None

Result Date

Example:

```
Public Sub OnResetAlarm(bRet As Boolean)
        Debug.Print DateTimeAck
End Sub
```

DateTimeACKMs, AlarmThresholdCmdTarget Property

Syntax DateTimeACKMs = _Integer

Description This property sets or returns the milliseconds of the second in which the alarm acknowledgement took place. To get the complete time data format you can combine it together with the DateTimeAck but be careful when using different variable types (date format and integer for this property). This property changes the value displayed in the alarms window only and does not influence the historical log.

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub OnResetAlarm(bRet As Boolean)
        Debug.Print "ON = " & Format(DateTimeOn,"yyyy/mm/dd hh.nn.ss") & "," & DateTimeOnMs
            ' ON = 2001/03/16 11.27.17,10
        Debug.Print "ACK = " & Format(DateTimeAck,"yyyy/mm/dd hh.nn.ss") & "," &
        DateTimeACKMs   ' ACK = 2001/03/16 11.27.24,210
End Sub
```

DateTimeFromTimeStamp, AlarmThresholdCmdTarget Property

Syntax DateTimeFromTimeStamp = _Boolean

Description This property sets or returns the activation status of the alarm threshold's "Use Variable TimeStamp" property through which you can make the alarm's TimeStamp coincide with the that of the variable's.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim objAlarm As AlarmCmdTarget
        Dim objAlarmThreshold As AlarmThresholdCmdTarget
        Set objAlarm = GetAlarm("Alarm01")
        Set objAlarmThreshold = objAlarm.GetAlarmThreshold("Threshold01")
        MsgBox " DateTimeFromTimeStamp = " &
        CStr(objAlarmThreshold.DateTimeFromTimeStamp),vbInformation,GetProject
        Title
        Set objAlarm = Nothing
        Set objAlarmThreshold = Nothing
End Sub
```

DateTimeOFF, AlarmThresholdCmdTarget Property

Syntax DateTimeOFF = _Date

Description This property sets or returns the date and time in which the alarm is turned OFF. Accepts a date parameter type. This property changes the

value displayed in the alarms window only and does not influence the historical log.

Parameter	Description
None	None

Result Date

Example:

```
Public Sub OnResetAlarm(bRet As Boolean)
        DateTimeOFF = CDate(Date)
        Debug.Print DateTimeOFF
End Sub
```

DateTimeOFFMs, AlarmThresholdCmdTarget Property

Syntax DateTimeOFFMs = _Integer

Description This property sets or returns the milliseconds of the second in which the alarm turned OFF. To get a complete time format you can combine this with the DateTimeOFF taking care with the different variables being used (date format and integer for this property). This property changes the value displayed in the alarms window only and does not influence the historical log.

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub OnResetAlarm(bRet As Boolean)
        Debug.Print "OFF = " & Format(DateTimeOff,"yyyy/mm/dd hh.nn.ss") & "," &
        DateTimeOFFMs
End Sub
```

DateTimeON, AlarmThresholdCmdTarget Property

Syntax DateTimeON = _Date

Description This property sets or resets the data and time in which the alarm turned ON. Accepts a date parameter. This changes the value displayed in the alarms window only and does not influence the historical log.

Parameter	Description
None	None

Result Date

Example:

```
Public Sub OnAckAlarm(bRet As Boolean)
        DateTimeOn = CDate(Date)
        Debug.Print DateTimeOn
End Sub
```

DateTimeOnMs, AlarmThresholdCmdTarget Property

Syntax DateTimeOnMs = _Integer

Description This property sets or returns the milliseconds of the secon in which the alarm turned ON. To get the complete time format you can combine this with the DateTimeOn taking care with the different variables being used (date format and integer). This property changes the value displayed in the alarms window only and has no influence on the historical log.

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub OnResetAlarm(bRet As Boolean)
        Debug.Print "ON = " & Format(DateTimeOn,"yyyy/mm/dd hh.nn.ss") & "," &
        DateTimeOnMs      ' ON = 2001/03/16 11.27.17,10
        Debug.Print "ACK = " & Format(DateTimeAck,"yyyy/mm/dd hh.nn.ss") & "," &
        DateTimeAckMs  ' ACK = 2001/03/16 11.27.24,210
End Sub
```

DateTimeRESET, AlarmThresholdCmdTarget Property

Syntax DateTimeRESET = _Date

Description This property sets or returns the date and time in which the alarm reset took place. Accepts a date parameter. This property changes the value displayed in the alarms window only and does not influence the historical log.

Parameter	Description

None	None

Result Date

Example:
```
Public Sub OnResetAlarm(bRet As Boolean)
        Debug.Print DateTimeRESET
End Sub
```

DateTimeRESETMs, AlarmThresholdCmdTarget Property

Syntax DateTimeRESETMs = _Integer

Description This property sets or returns the milliseconds of the second in which the alarm reset took place. To get a complete date format you can combine this with the DateTimeAck taking care with the different variables being used (date format and integer for this property). This property changes the value displayed in the alarms window only and does not influence the historical log.

Parameter	Description
None	None

Result Integer

Example:
```
Public Sub OnResetAlarm(bRet As Boolean)
        Debug.Print "RESET = " & Format(DateTimeRESET,"yyyy/mm/dd hh.nn.ss") & "," &
         DateTimeRESETMs
End Sub
```

DurationFormat, AlarmThresholdCmdTarget Property

Syntax DurationFormat = _String

Description This property allows a message to be inserted which will be filed under the "CommCol" column in the "Alarms" table of the Historical Log. The message will be recorded only on the "Alarm Off" event.
The message can include the following special codes only:

- **%D** = Days of alarm duration
- **%H** = Hours of alarm duration
- **%M** = Minutes of alarm duration
- **%S** = Seconds of alarm duration

When the entry field is left empty, Movicon will automatically insert the

alarm's total duration with the following string:

Duration total 0,00:00:00

where 00:00:00 indicates the alarm duration in days, hours, minutes and seconds.

Parameter	Description
None	None

Result String

Example:

```
Public Sub OnResetAlarm(bRet As Boolean)
        Debug.Print DurationFormat
End Sub
```

Help, AlarmThresholdCmdTarget Property

Syntax Help = lpar

Description This property sets or returns the ID of the string selected for the alarm help. Accepts a string type parameter.

Parameter	Description
lpar As String	ID of alarm's help string

Result String

Example:

```
Public Sub AlarmLoading()
        Help = "STR00001"
        Debug.Print Help 'Return-> STR00001
End Sub
```

LastComment, AlarmThresholdCmdTarget Property

Syntax LastComment = _String

Description This property allows a comment for the alarm in question to be read or written. This comment will be the same one that can be inserted or read using the Alarm Window 'Comment Button'.

Parameter	Description

None	None

Result String

Example:

```
Public Sub Click()
        Dim objAlarm As AlarmCmdTarget
        Dim objAlarmThreshold As AlarmThresholdCmdTarget
        Set objAlarm = GetAlarm("Alarm01")
        Set objAlarmThreshold = objAlarm.GetAlarmThreshold("Threshold01")
        MsgBox " LastComment = " & CStr(objAlarmThreshold.LastComment
        ),vbInformation,GetProjectTitle
        Set objAlarm = Nothing
        Set objAlarmThreshold = Nothing
End Sub
```

LastTotalTimeOn, AlarmThresholdCmdTarget Property

Syntax LastTotalTimeOn

Description This property returns a date type value indicating the time in which the alarm turned ON for the last time. When the alarm turns OFF this value will be zeroed.

Parameter	Description
None	None

Result Date

Example:

```
Public Sub Click()
        Dim objAlarm As AlarmCmdTarget
        Dim objAlarmThreshold As AlarmThresholdCmdTarget
        Set objAlarm = GetAlarm("Alarm01")
        Set objAlarmThreshold = objAlarm.GetAlarmThreshold("Threshold01")
        MsgBox " LastTotalTimeOn= " &
        CStr(objAlarmThreshold.LastTotalTimeOn),vbInformation,GetProjectTitle
        Set objAlarm = Nothing
        Set objAlarmThreshold = Nothing
End Sub
```

Log, AlarmThresholdCmdTarget Property

Syntax Log = _Boolean

Description This property allows you to specify whether to activate or deactivate the recording function in Historical Log of the alarm or message when the relative events occur (ON, OFF, ACK, RESET). These recordings will be contained in the "Alarms" table of the Historical Log Database created by

Movicon in the defined format or in the Alarms.dat file,depending on the data source you are using (ODBC or IMDB).

If you are using the ODBC connection, the Historical Log is created with the Movicon default settings, but the Historical Log file can be customized when put into use, where you can create a personalized ODBC link and define a different table name. These functionalities can be carried out from the Project's "Historical Log Settings".

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub AlarmLoading()
        Debug.Print Log
End Sub
```

Name, AlarmThresholdCmdTarget Property

Syntax Name

Description This property returns the threshold name. This property is in read only.

Parameter	Description
None	None

Result String

Example:

```
Public Sub AlarmLoading()
        Debug.Print Name
End Sub
```

PlaysoundContinuously, AlarmThresholdCmdTarget Property

Syntax PlaysoundContinuously = _Boolean

Description This property allows you to set the behaviour of the sound file associated to the alarm. When this property is activated the file will be executed continuously until the alarm is silenced. Otherwise the file will be executed once only on alarm occurrence.

Parameter	Description
None	None

Result Boolean

Example:
Public Sub AlarmLoading()
Debug.Print **PlaysoundContinuously**
End Sub

Print, AlarmThresholdCmdTarget Property

Syntax Print = _Boolean

Description This property allows you to set the print function of the alarm or message to activate or deactivate when the relevant events occur (ON, OFF, ACK, RESET).
The print function must be activated and configured in the "Historical Log Print" settings beforehand.

Parameter	Description
None	None

Result Boolean

Example:
Public Sub AlarmLoading()
Debug.Print **Print**
End Sub

ReadAccessLevel, AlarmThresholdCmdTarget Property

Syntax ReadAccessLevel = _Long

Description This property sets or returns the Access Level mask needed for displaying the alarms in the Alarm Window. When the Access Level mask of the user logged on at that moment does not correspond with that set in the control, the user will not be able to view the alarms. The "0000" and "FFFF" levels render the object accessible in read to any user. The logging of alarms will naturally be executed independently from the user's access rights logged on. For further details on "Access Levels" please refer to the paragraph on"User Levels and Access Levels".

Parameter	Description

None	None

Result Long

Example:
```
Public Sub AlarmLoading()
        Debug.Print ReadAccessLevel
End Sub
```

Recipient, AlarmThresholdCmdTarget Property

Syntax Recipient = _String

Description By using this property you can retrieve or set the recipient user or user group to which messages, SMS, E-mails etc., are to be sent.

Parameter	Description
None	None

Result String

Example:
```
Public Sub AlarmLoading()
        Debug.Print Recipient
End Sub
```

RepeatSpeechEverySec, AlarmThresholdCmdTarget Property

Syntax RepeatSpeechEverySec = _Long

Description This property allows you to set or display the time expressed in seconds after which the alarm's speech must be repeated. When setting this property to the "0" value the alarm's speech will be repeated only once. Wwhen you set the time here, you must take into account the time needed to execute the alarm speech.

Parameter	Description
None	None

Result Long

Example:
```
Public Sub AlarmLoading()
        Debug.Print RepeatSpeechEverySec
End Sub
```

SecDelay, AlarmThresholdCmdTarget Property

Syntax SecDelay = _Long

Description This property sets or returns the alarm's delay time. You can set a numeric value between 0 and 65535. This value, expressed in seconds, sets the delay time of the alarm intervention, creating a filter on the threshold.
The default value is zero (no delay).

Parameter	Description
None	None

Result Long

Example:
```
Public Sub AlarmLoading()
        Debug.Print SecDelay
End Sub
```

SendFaxEnabledACK, AlarmThresholdCmdTarget Property

Syntax SendFaxEnabledACK = _Boolean

Description This property activates or deactivates the management for sending Faxes. This management requires that the appropriate functions for sending faxes and any modem be enabled and configured beforehand.
The message will be sent upon alarm acknowledgement (ACK).

Parameter	Description
None	None

Result Boolean

Example:
```
Public Sub AlarmLoading()
        Debug.Print SendFaxEnabledACK
End Sub
```

SendFaxEnabledOFF, AlarmThresholdCmdTarget Property

Syntax SendFaxEnabledOFF = _Boolean

Description This property activates or deactivates the management for sending faxes. This management requires that the appropriate functions for sending faxes and any modem be enabled and configured beforehand.
The message will be sent upon alarm OFF, independently from the acknowledge or reset status.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub AlarmLoading()
        Debug.Print SendFaxEnabledOFF
End Sub
```

SendFaxEnabledON, AlarmThresholdCmdTarget Property

Syntax SendFAXEnabledON= _Boolean

Description This property activates or deactivates the management for sending faxes. This management requires that the appropriate functions for sending faxes and any modem be enabled and configured beforehand.
The message will be sent upon alarm ON intervention.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub AlarmLoading()
        Debug.Print SendFAXEnabledON
End Sub
```

SendFaxEnabledRESET, AlarmThresholdCmdTarget Property

Syntax SendFaxEnabledRESET = _Boolean

Description This property activates or deactivates the management for sending faxes. This management requires that the appropriate functions for sending faxes and any modem be enabled and configured beforehand.
The message is sent the moment the alarm is RESET.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub AlarmLoading()
        Debug.Print SendFaxEnabledRESET
End Sub
```

SendMailEnabledACK, AlarmThresholdCmdTarget Property

Syntax SendMailEnabledACK = _Boolean

Description This property activates or deactivates the management for sending E-mail messages. This management requires that the E-mail post manager be installed both for the Client and the Server as well as the right modem and Internet connection.
The message will be sent the moment in which the Alarm is acknowledged (ACK).

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub AlarmLoading()
        Debug.Print SendMailEnabledACK
End Sub
```

SendMailEnabledOFF, AlarmThresholdCmdTarget Property

Syntax SendMailEnabledOFF = _Boolean

Description This property activates or deactivates the management for sending E-mail messages. This management requires that the E-mail post manager be installed both for the Client and the Server as well as the right modem and Internet connection.
The message will be sent the moment in which the Alarm is turned OFF, independently from the acknowledge and reset status.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub AlarmLoading()
        Debug.Print SendMailEnabledOFF
End Sub
```

SendMailEnabledON, AlarmThresholdCmdTarget Property

Syntax SendMailEnabledON = _Boolean

Description This property activates or deactivates the management for sending E-mail messages. This management requires that the E-mail post manager be installed both for the Client and the Server as well as the right modem and Internet connection.
The message will be sent the moment in which the Alarm is turned ON.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub AlarmLoading()
        Debug.Print SendMailEnabledON
End Sub
```

SendMailEnabledRESET, AlarmThresholdCmdTarget Property

Syntax SendMailEnabledRESET = _Boolean

Description This property activates or deactivates the management for sending E-mail messages. This management requires that the E-mail post manager be installed both for the Client and the Server as well as the right modem and Internet connection.
The message will be sent the moment in which the Alarm is RET.

Parameter	Description
None	None

Result Boolean

Example:
Public Sub AlarmLoading()
Debug.Print **SendMailEnabledRESET**
End Sub

SendSMSEnabledACK, AlarmThresholdCmdTarget Property

Syntax SendSMSEnabledACK = _Boolean

Description This property activated or deactivated the management for sending SMS messages. This management requires that the appropriate functions sending SMS and any need of a modem be enabled and configured.
The message will be sent the moment in which the alarm is acknowledged (ACK).

Parameter	Description
None	None

Result Boolean

Example:
Public Sub AlarmLoading()
Debug.Print **SendSMSEnabledACK**
End Sub

SendSMSEnabledOFF, AlarmThresholdCmdTarget Property

Syntax SendSMSEnabledOFF = _Boolean

Description This property activated or deactivated the management for sending SMS messages. This management requires that the appropriate functions sending SMS and any need of a modem be enabled and configured.
The message will be sent the moment in which the alarm is turned OFF, independently from the acknowledge or reset status.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub AlarmLoading()
        Debug.Print SendSMSEnabledOFF
End Sub
```

SendSMSEnabledON, AlarmThresholdCmdTarget Property

Syntax SendSMSEnabledON = _Boolean

Description This property activated or deactivated the management for sending SMS messages. This management requires that the appropriate functions sending SMS and any need of a modem be enabled and configured.
The message will be sent the moment in which the alarm is turned ON.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub AlarmLoading()
        Debug.Print SendSMSEnabledON
End Sub
```

SendSMSEnabledRESET, AlarmThresholdCmdTarget Property

Syntax SendVoiceEnabledRESET = _Boolean

Description This property activated or deactivated the management for sending SMS messages. This management requires that the appropriate functions sending SMS and any need of a modem be enabled and configured.
The message will be sent the moment in which the alarm is RESET.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub AlarmLoading()
        Debug.Print SendVoiceEnabledRESET
End Sub
```

SendVoiceEnabledACK, AlarmThresholdCmdTarget Property

Syntax SendVoiceEnabledACK = _Boolean

Description This property activates or deactivates the management for sending Vocal messages. This management requires that the appropriate functions be enabled for sending vocal messages (TAPI), and that the desired phonemes are installed and any modem is appropriately configured.
The message will be sent the moment in which the alarm is acknowledged (ACK).

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub AlarmLoading()
        Debug.Print SendVoiceEnabledACK
End Sub
```

SendVoiceEnabledOFF, AlarmThresholdCmdTarget Property

Syntax SendVoiceEnabledOFF = _Boolean

Description This property activates or deactivates the management for sending Vocal messages. This management requires that the appropriate functions be enabled for sending vocal messages (TAPI), and that the desired phonemes are installed and any modem is appropriately configured.
The message will be sent the moment in which the alarm is turned OFF, independently from the acknowledge or reset status.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub AlarmLoading()
        Debug.Print SendVoiceEnabledOFF
End Sub
```

SendVoiceEnabledON, AlarmThresholdCmdTarget Property

Syntax SendVoiceEnabledON = _Boolean

Description This property activates or deactivates the management for sending Vocal messages. This management requires that the appropriate functions be enabled for sending vocal messages (TAPI), and that the desired phonemes are installed and any modem is appropriately configured.
The message wil be sent the moment in which the alarm is turned ON.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub AlarmLoading()
        Debug.Print SendVoiceEnabledON
End Sub
```

SendVoiceEnabledRESET, AlarmThresholdCmdTarget Property

Syntax SendSMSEnabledRESET = _Boolean

Description This property activates or deactivates the management for sending Vocal messages. This management requires that the appropriate functions be enabled for sending vocal messages (TAPI), and that the desired phonemes are installed and any modem is appropriately configured.
The message will be send the moment in which the alarm is RESET.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub AlarmLoading()
        Debug.Print SendSMSEnabledRESET
End Sub
```

Severity, AlarmThresholdCmdTarget Property

Syntax Severity = _Long

Description This property sets or returns the severity desired for the alarm. The severity will be displayed and recorded in the purpose-built field reversed for the alarm. A number between 0 and 65535 can be assigned. The assigned severity number will be recorded in the alarms widow and the Historical Log.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub AlarmLoading()
        Debug.Print Severity
End Sub
```

SpeechEnabled, AlarmThresholdCmdTarget Property

Syntax SpeechEnabled = _Boolean

Description This property allows you to enable the speech functions for vocal synthesis of the text strings that the alarm is built with. The alarm's text will then be spoken by Movicon when they occur.

In order to activate this function you need to install the Microsoft API speeching engine beforehand (or any other brand name), which is not usually installed automatically in Windows. The "Speech" folder on the Movicon CD contains the auto-installation of the Microsoft speech engine.
In addition to installing the speech engine you also need to install the phonemics of the desired language. The phonemics depend on the language being used and are normally supplied separately. Therefore this job is left to the user to get hold of and install the right ones desired.

Parameter	Description
None	None

Result Boolean

Example:
```
Public Sub AlarmLoading()
        Debug.Print SpeechEnabled
End Sub
```

SpeechEnableVariable, AlarmThresholdCmdTarget Property

Syntax SpeechEnableVariable = String

Description This property sets or returns the name of the variable which manages the Speech enabling (alarm threshold "Enabling Speech Variable" style property).

Parameter	Description
None	None

Result String

Example:
```
Public Sub AlarmLoading()
        Debug.Print SpeechEnableVariable
End Sub
```

Status, AlarmThresholdCmdTarget Property

Syntax Status = _Integer

Description This property sets or returns the status of the alarm.

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub OnResetAlarm(bRet As Boolean)
        Debug.Print Status
End Sub
```

StatusVariable, AlarmThresholdCmdTarget Property

Syntax StatusVariable = _String

Description This property sets or returns the name of the alarm threshold's Status-Command variable. When setting a bit type variable it will only be possible to command the alarm's acknowledgement. Setting a byte type variable (or with a higher number of bits) it will also be possible to get the alarm's status or command its reset.

Parameter	Description
None	None

Result String

Example:

```
Public Sub OnResetAlarm(bRet As Boolean)
        Debug.Print "ON = " & Format(DateTimeOn,"yyyy/mm/dd hh.nn.ss") & "," &
        DateTimeOnMs     ' ON = 2001/03/16 11.27.17,10
        Debug.Print "ACK = " & Format(DateTimeAck,"yyyy/mm/dd hh.nn.ss") & "," &
        DateTimeAckMs  ' ACK = 2001/03/16 11.27.24,210
End Sub
```

SupportAcknowledge, AlarmThresholdCmdTarget Property

Syntax SupportAcknowledge = _Boolean

Description By using this property you can set the activation or deactivation of the alarm's Acknowledge (ACK) function.

Parameter	Description
None	None

Result Boolean

Example:
```
Public Sub AlarmLoading()
        Debug.Print Attachment
End Sub
```

SupportReset, AlarmThresholdCmdTarget Property

Syntax SupportReset = _Boolean

Description By using this property you can set the activation or deactivation of the alarm's RESET function.

Parameter	Description
None	None

Result Boolean

Example:
```
Public Sub AlarmLoading()
        Debug.Print SupportReset
End Sub
```

TextColor, AlarmThresholdCmdTarget Property

Syntax TextColor = _Long

Description This property sets or returns the color of the text relating to the alarm or message displayed in the appropriate window.

Parameter	Description
None	None

Result Long

Example:
```
Public Sub AlarmLoading()
        Debug.Print TextColor
End Sub
```

SupportResetWithConditionOn, AlarmThresholdCmdTarget Property

Syntax SupportReset = _Boolean

Description This property is used for activating or not the alarm's reset function (RESET).

Parameter	Description
None	None

Result Boolean

Example:
```
Public Sub AlarmLoading()
        Debug.Print SupportReset
End Sub
```

Text, AlarmThresholdCmdTarget Property

Syntax Text = _String

Description This property sets or returns the ID of the string selected for the alarm's text.

Parameter	Description
None	None

Result String

Example:
```
Public Sub AlarmLoading()
        Text = "STR00001"
        Debug.print Text 'STR00001
End Sub
```

Threshold, AlarmThresholdCmdTarget Property

Syntax Threshold = _Double

Description This property sets or returns the minimum threshold value for alarm activation. This value is only consider when "Betweem" has been selected as "Activation Condition". Otherwise it will have no effect whatsoever.

Parameter	Description
None	None

Result Double

Example:

```
Public Sub AlarmLoading()
        Msgbox "Threshold is: " & cstr(ThresholdLow), vbInformation, GetProjectTitle
End Sub
```

ThresholdLow, AlarmThresholdCmdTarget Property

Syntax Threshold = _Double

Description This property sets or returns the minimum threshold value for the alarm activation. This value is only considered when "Between" has been selected as "Activation Condition". This value will have no effect in other cases.

Parameter	Description
None	None

Result Double

Example:

```
Public Sub AlarmLoading()
        Msgbox "Threshold is: " & cstr(ThresholdLow), vbInformation, GetProjectTitle
End Sub
```

TotalTimeOn, AlarmThresholdCmdTarget Property

Syntax TotalTimeOn

Description This property returns a double value type indicating the total time, expressed in seconds, in which the the alarm remained active in ON status.

Parameter	Description
None	None

Result Double

Example:

```
Public Sub Click()
        Dim objAlarm As AlarmCmdTarget
        Dim objAlarmThreshold As AlarmThresholdCmdTarget
        Set objAlarm = GetAlarm("Alarm01")
        Set objAlarmThreshold = objAlarm.GetAlarmThreshold("Threshold01")
        MsgBox " TotalTimeOn = " &
        CStr(objAlarmThreshold.TotalTimeOn),vbInformation,GetProjectTitle
        Set objAlarm = Nothing
        Set objAlarmThreshold = Nothing
End Sub
```

VariableSeverity, AlarmThresholdCmdTarget Property

Syntax VariableSeverity = _String

Description This property returns or set the name of the variable which defines the alarm threshold's Priority value.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim objAlarm As AlarmCmdTarget
        Dim objAlarmThreshold As AlarmThresholdCmdTarget

        Set objAlarm = GetAlarm("AlarmNoReset")
        Set objAlarmThreshold = objAlarm.GetAlarmThreshold("NoReset")

        MsgBox " VariableSeverity = " & CStr(objAlarmThreshold.VariableSeverity
         ),vbInformation,GetProjectTitle
```

```
        Set objAlarm = Nothing
        Set objAlarmThreshold = Nothing
End Sub
```

VariableThreshold, AlarmThresholdCmdTarget Property

Syntax VariableThreshold = _String

Description This property sets or returns the name of the variable whose value is used instead of the Threshold property ('Value'). In this way the threshold is made dynamic. When a nothing string is inserted, Movicon will consider the fixed threshold only.

Parameter	Description
None	None

Result String

Example:

```
Public Sub AlarmLoading()
        Debug.Print "Threshold  is " & Threshold
End Sub
```

VariableThresholdLow, AlarmThresholdCmdTarget Property

Syntax VariableThresholdLow = _String

Description This property sets or returns the name of the variable whose value is used instead of the ThresholdLow property ('Minimum Activation Valor'). In this way the threshold can be made dynamic. When inserting a null string, Movicon will only take the fixed threshold into consideration.

Parameter	Description
None	None

Result Double

Example:

```
Public Sub AlarmLoading()
        Msgbox "Variable Threshold Low Name is: " & cstr(VariableThresholdLow),
vbInformation, GetProjectTitle
End Sub
```

WriteAccessLevel, AlarmThresholdCmdTarget Property

Syntax WriteAccessLevel = _Long

Description This property sets or returns the Access Level mask so that the alarm's acknowledge and reset can be exectued. When the Access level mask of the user logged on in that moment does not correspond to that set on the control , the user will not be able to carry out any operations on the commands associated to the alarm. The "FFF" level makes the object accessible in write to any user.
For further information on the "Access Levels" please refer to the paragraph titled "User Levels and Access Levels".

Parameter	Description
None	None

Result Long

Example:

```
Public Sub AlarmLoading()
        Debug.Print WriteAccessLevel
End Sub
```

6. AlarmWndCmdTarget

OnAckAll, AlarmWndCmdTarget Event

Description This event is notified each time a request is made to acknowledge all alarms displayed in the window.

This function is not supported by the "Alarm Banner" object.

Parameter	Description
bRet As Boolean	Enabling on alarm acknowledgement. When set at False this event will not acknowledge the alarms.

OnAckSel, AlarmWndCmdTarget Event

Description Event notified each time a request to acknowledge the alarms selected in the display window.

This function is not supported by the "Alarm Banner" object.

Parameter	Description
bRet As Boolean	Enabling on acknowledging selected alarm. When set at False this event will not acknowledge the Alarm.

OnGetHistory, AlarmWndCmdTarget Event

Description Event notified each time a request using the "Get History" button to load the history of an alarm.

Parameter	Description
bRet As Boolean	Enabling at data retrieval. When set at False this event will not return the alarm's history.

OnHelp, AlarmWndCmdTarget Event

Description Event occurs each time a request is made to display the help file associated to the selected alarm.

Parameter	Description
bRet As Boolean	Enabling upon opening of Help file. When set at False, this event will not open the help file.

OnCommentSel, AlarmWndCmdTarget Event

Description Event notified each time a request to enter a comment is made for the alarm selected in the display window.

This function is not managed by the "Alarm Banner" object.

Parameter	Description
bRet As Boolean	Enabled when comment is inserted. When set to False, this event will cancel the comment request.

OnInsertOrUpdateAlarm, AlarmWndCmdTarget Event

Description Event occurs each time a new alarm is inserted or when the status of the alarm, existing in the display window, is modified.

Parameter	Description
AlarmObject As Object	Alarm Object Inserted or status changed.
bRet As Boolean	Enabling upon inserting an alarm into window. When set at False, this event will not notify window of insertion of new alarm event or the status change event of already existing alarm.

OnOPCAEEvent, AlarmWndCmdTarget Event

Description Event occurs each time an alarm event via OPC occurs.

This event is not supported in Windows CE.

Parameter	Description
zSource As String	Event source
dDate As Date	Date and time
szEvent As String	Event type
dwSeverity As Long	Severity level

bRet As Boolean	Enabling upon the OPC event in the window. When set to False, the OPC event will not be notified to the Alarms window.

OnResetAll, AlarmWndCmdTarget Event

Description Event occurs each time a request is made for resetting all the alarms in the display window.

This function is not supported by the "Alarm Banner" object.

Parameter	Description
bRet As Boolean	Enabling at alarm resets. When set at False, this event will not reset alarms.

OnResetSelSel, AlarmWndCmdTarget Event

Description Event occurs each time a request is made for resetting the alarm in the display window.

This function is not supported by the "Alarm Banner" object.
.

Parameter	Description
bRet As Boolean	Enabling at selected alarm reset. When set to False, this event will not reset the alarm.

OnToggleSound, AlarmWndCmdTarget Event

Description Event occurs each time a request is made to silence the sound associated to the live alarms.

Parameter	Description
bRet As Boolean	Enabling on Toggle Sound command execution. When set at False, this event will not execute command.

AckSelectedAlarms, AlarmWndCmdTarget Function

Syntax AckSelectedAlarms()

Description This function permits the acknowledgment of all the alarms selected.

This function is not managed by the "Alarm Banner" object. Always returns False.

Parameter	Description
None	None

Result Boolean

Example:
Sub Click()
AckSelectedAlarms
End Sub

EditLayout, AlarmWndCmdTarget Function

Syntax EditLayout()

Description This function opens the configuration window of fields to be displayed in the Alarm Window.

This function is only executed if the "Show Control window" property has been enabled in the Window object. Otherwise the "Field Choice Window" will not open and this function will return the "False" value.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result Boolean

Example:
Sub Click()
EditLayout
End Sub

EditCopy, AlarmWndCmdTarget Function

Syntax EditCopy()

Description This property executes a copy of the selected alarm contents to the clipboard.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result Boolean

Example:

```
Dim objAlarmWnd As AlarmWndCmdTarget
Public Sub Click()
        Debug.Print objAlarmWnd.EditCopy
End Sub

Public Sub SymbolLoading()
        Set objAlarmWnd = GetSynopticObject.GetSubObject("ALRWindow").GetObjectInterface
End Sub
```

GetNumTotalAlarms, AlarmWndCmdTarget Function

Syntax GetNumTotalAlarms()

Description This function returns the number or alarms presented in the Alarm Window.

This function is not managed by the "Alarm Banner" object. Always returns 0.

Parameter	Description
None	None

Result Long

Example:

```
Sub Click()
        Dim lResult As Long
        lResult = GetNumTotalAlarms
        Debug.Print lResult
End Sub
```

GetSelectedAlarm, AlarmWndCmdTarget Function

Syntax GetSelectedAlarm()

Description This function returns the selected alarm.

This function is not managed by the "alarm Banner". always returns a nothing object.

Parameter	Description
None	None

Result Object
If Function has been executed successfully it will retrieve an object of type AlarmThresholdCmdTarget if otherwise Nothing is returned.

Example:

```
Sub Click()
        Dim objAlarm As AlarmThresholdCmdTarget
        Set objAlarm = GetSelectedAlarm
        Debug.Print objAlarm.Condition
        Set objAlarm = Nothing
End Sub
```

GetSelHistory, AlarmWndCmdTarget Function

Syntax GetSelHistory()

Description This method executes the same command relating to the "Get History" button, meaning that it retrieves historical information for the selected alarm. When function's return value is False, this means that there is an error in the operation.

This function is not managed by the "alarm Banner

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim objWnd As AlarmWndCmdTarget
        Set objWnd = GetSynopticObject.GetSubObject("AlarmWnd").GetObjectInterface
        If objWnd Is Nothing Then Exit Sub
        objWnd.GetSelHistory
        Set objWnd = Nothing
End Sub
```

LoadExtSettings, AlarmWndCmdTarget Function

Syntax LoadExtSettings

Description This function permits the object's relating external file settings to be loaded. This file can be specified in design mode in the "External File settings" property or in the "ExtSettingsFile" interface properties. The extension provided for this file is ".SXML".

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
Dim objSymbol As AlarmWndCmdTarget
Set objSymbol = GetSynopticObject.GetSubObject("TestObject").GetObjectInterface
If objSymbol Is Nothing Then Exit Sub
objSymbol.ExtSettingsFile = "test.sxml"
objSymbol.LoadExtSettings
Set objSymbol = Nothing
End Sub
```

RecalcLayout, AlarmWndCmdTarget Function

Syntax RecalcLayout()

Description This function resizes the columns of the display window according to the sizes which have been set for each column.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Click()
        Dim bResult As Boolean
        bResult = RecalcLayout
        Debug.Print bResult
End Sub
```

Refresh, AlarmWndCmdTarget Function

Syntax Refresh()

Description This function carries out a refresh of the object's graphics. You need to use this function for instance, after a property has been changed to add or take away columns from the alarm window.

Parameter	Description
None	None

Result Boolean

Example:
Sub Click()
Refresh
End Sub

ResetSelectedAlarms, AlarmWndCmdTarget Function

Syntax ResetSelectedAlarms()

Description This function permits the reset of the of the selected alarms.

This function is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result Boolean

Example:
Sub Click()
ResetSelectedAlarms
End Sub

SaveExtSettings, AlarmWndCmdTarget Function

Syntax SaveExtSettings

Description This function permits the objects settings to be save in the relating external settings file. This file can be specified when in design mode in the "Ext. Settings File" property, or using the property from the "ExtSettingsFile" interface. The extension provided for this file is ".SXML".

Parameter	Description
None	None

Result Long

Example:
```
Public Sub Click()
Dim objSymbol As AlarmWndCmdTarget
Set objSymbol = GetSynopticObject.GetSubObject("TestObject").GetObjectInterface
If objSymbol Is Nothing Then Exit Sub
objSymbol.ExtSettingsFile = "test.sxml"
```

```
objSymbol.SaveExtSettings
Set objSymbol = Nothing
End Sub
```

SelectAll, AlarmWndCmdTarget Function

Syntax SelectAll()

Description This function permits all the alarms displayed in the Alarm Window to be selected. Returns the number of alarms selected.

This function is not managed by the "Alarm Banner" object. Always returns as 0.

Parameter	Description
None	None

Result Long

Example:

```
Sub Click()
        SelectAll
End Sub
```

AckAllBtnText, AlarmWndCmdTarget Property

Syntax AckAllBtnText = _String

Description This property returns the text which has to appear on the "Acknowledge All Button". When the field is left empty, the default text will be used instead.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result String

Example:

```
Sub Click()
        Debug.Print AckAllBtnText
End Sub
```

AckSelBtnText, AlarmWndCmdTarget Property

Syntax AckSelBtnText = _String

Description This property returns the text which has to appear on the "Acknowledge Sel Button". When the field is left empty the default text will be used instead.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result String

Example:

```
Sub Click()
        Debug.Print AckSelBtnText
End Sub
```

AlarmFilter, AlarmWndCmdTarget Property

Syntax AlarmFilter = _String

Description This property permits a filter to be inserted for displaying alarms according to their texts. The filter is applied to the "Alarm Description" column and can contain one or more of the "*" special characters (ie. *Turbine*).

Parameter	Description
None	None

Result String

Example:

```
Sub Click()
        Debug.Print AlarmFilter
End Sub
```

AlarmFilterMask, AlarmWndCmdTarget Property

Syntax AlarmFilterMask = _Integer

Description This property permits a filter to be inserted according to the status of alarms to be displayed in the window.

The filter can obtain the follwoing values:
1 = Alarm ON
2 = Alarm ACK
4 = Alarm OFF ACK
8 = Alarm OFF
16 = No Alarm

For further information see Alarm Mask Filter.

Parameter	Description
None	None

Result Integer

Example:

```
Sub Click()
        Debug.Print AlarmFilter
End Sub
```

AlarmFilterSeverity, AlarmWndCmdTarget Property

Syntax AlarmFilterSeverity = _Long

Description This property sets or returns the severity priority used for executing the filter in the Alarms Window.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim objWnd As AlarmWndCmdTarget

        Set objWnd = GetSynopticObject.GetSubObject("AlarmWnd").GetObjectInterface
        If objWnd Is Nothing Then Exit Sub
        MsgBox " AlarmFilterSeverity = " &
        cstr(objWnd.AlarmFilterSeverity),vbInformation,GetProjectTitle
        Set objWnd = Nothing
End Sub
```

AlarmFilterSeverityCondition, AlarmWndCmdTarget Property

Syntax AlarmFilterSeverityCondition = _Integer

Description This property sets or returns the condition type used for executing the filter by severity in the Alarms Window. The possible values are:

0 = Equal
1 = Minor-egual
2 = Major-equal

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        Dim objWnd As AlarmWndCmdTarget
        Set objWnd = GetSynopticObject.GetSubObject("AlarmWnd").GetObjectInterface
        If objWnd Is Nothing Then Exit Sub
        MsgBox " AlarmFilterSeverityCondition = " &
        cstr(objWnd.AlarmFilterSeverityCondition),vbInformation,GetProjectTitle
        Set objWnd = Nothing
End Sub
```

AreaFilter, AlarmWndCmdTarget Property

Syntax AreaFilter = _String

Description This property permits you to insert a filter for displaying the alarms belonging to a certain area only. The filter can contain one or more special "*" characters (ie. *Area*).

Parameter	Description
None	None

Result String

Example:

```
Sub Click()
        Debug.Print AreaFilter
End Sub
```

AutoLayout, AlarmWndCmdTarget Property

Syntax AutoLayout = _Boolean

Description when this property is enabled, the layout will be set to automatic mode. This means that the columns will be automatically resized so that they all become visible within the area of the Alarm Window. When this property is disabled the columns will have the sizes setup in the programming stage when the window opens, with the possibility that the last ones on the right will not be visible unless the horizontal scroll bar is used to view them.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Click()
        Debug.Print AutoLayout
End Sub
```

Autoscroll, AlarmWndCmdTarget Property

Syntax Autoscroll = _Boolean

Description When this property is enabled, the active alarms scroll will be set to automatic mode. When this property is disabled only the manual scroll will be possible.

This property is not managed by the "Alarm Window". Always returns False.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim objAlarmBanner As AlarmWndCmdTarget
        Set objAlarmBanner =
        GetSynopticObject.GetSubObject("objAlarmBanner").GetObjectInterface
        objAlarmBanner.Autoscroll = Not objAlarmBanner.Autoscroll
        Set objAlarmBanner = Nothing
End Sub
```

BlinkTime, AlarmWndCmdTarget Property

Syntax BlinkTime = _Long

Description This property represents the Blink time for the alarms still not acknowledged. The value is expressed in milliseconds.

Parameter	Description
None	None

Result Long

Example:
```
Sub Click()
        Debug.Print BilnkTime
End Sub
```

ButtonPos, AlarmWndCmdTarget Property

Syntax ButtonPos

Description This setting returns the position in which the buttons must appear in the Alarm Window.

The options are:
0 = left
1 = top
2 = right
3 = bottom

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result Integer

Example:
```
Sub Click()
        ButtonPos = 2
        Debug.Print ButtonPos
End Sub
```

ButtonSize, AlarmWndCmdTarget Property

Syntax ButtonSize

Description This setting returns the size of the buttons which are to be displayed in the Alarm Window.

The options area:

0 = small
1 = medium
2 = large

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result Integer

Example:

```
Sub Click()
        ButtonSize = 2
        Debug.Print ButtonSize
End Sub
```

Clickable, AlarmWndCmdTarget Property

Syntax Clickable = _Boolean

Description This property allows you to establish whether the operator can interact with the Alarm Window. It will not be able to manage the control with the mouse or the keyboard when this property is disabled.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Click()
        Debug.Print Clickable
End Sub
```

ExtSettingsFile, AlarmWndCmdTarget Property

Syntax ExtSettingsFile = _String

Description This property sets or returns the external configuration file for the referenced object. the file can be alsow specified in design mode in the object's "Configuration File" property. The extension provided for this file is ".SXML".

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
Dim objSymbol As AlarmWndCmdTarget
Set objSymbol = GetSynopticObject.GetSubObject("TestObject").GetObjectInterface
If objSymbol Is Nothing Then Exit Sub
objSymbol.ExtSettingsFile = "test.sxml"
objSymbol.SaveExtSettings
Set objSymbol= Nothing
End Sub
```

FormatDateTime, AlarmWndCmdTarget Property

Syntax FormatDateTime = _String

Description This property allows you to insert the date and time format with which the time is to be displayed in the "Time ON", "Time Ack", "Time Off" and "Time Reset" columns. All the format codes that can be used in this property are listed in the Drawings and Controls Stile Properties section.

Parameter	Description
None	None

Result String

Example:

```
Sub Click()
        Debug.Print FormatDateTime
End Sub
```

FormatDuration, AlarmWndCmdTarget Property

Syntax FormatDuration = _String

Description This property permits you to insert the format to be used in the Alarm Window's "Duration" Column. All the format codes that can be used in this property are listed in the Drawings and Controls Stile Properties section.
The duration value will be updated only on the "Alarm Off" event.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result String

Example:

```
Sub Click()
        Debug.Print FormatDuration
End Sub
```

GetHistoryBtnTex, AlarmWndCmdTarget Property

Syntax GetHistoryBtnTex = _String

Description This property sets or returns the text for the 'Get History' button. When setting a nothing string the default text will be displayed.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim objWnd As AlarmWndCmdTarget
        Set objWnd = GetSynopticObject.GetSubObject("AlarmWnd").GetObjectInterface
        If objWnd Is Nothing Then Exit Sub
        MsgBox " GetHistoryBtnTex = " & cstr(objWnd.GetHistoryBtnTex
        ),vbInformation,GetProjectTitle
        Set objWnd = Nothing
End Sub
```

GraphicButtons, AlarmWndCmdTarget Property

Syntax GraphicButtons = _Boolean

Description When Enabling this property, the Alarm Window buttons are drawn using an icon instead of text. The text will instead be displayed as a tooltip when positioning the mouse on top of the button.

The tooltip is not managed in Windows CE versions.

This property is not managed by the 'Alarm Banner' object.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Click()
        GraphicButtons = True
        RecalcLayout
End Sub
```

HasSpin, AlarmWndCmdTarget Property

Syntax HasSpin = _Boolean

Description When enabling this property, the spin button will be displayed to be used for scrolling active alarms in the window. The spin button will display when Setting this property with the "True" value. It will not display when this property is set to "False".

This property is not managed by the "alarm Window". Always returns False.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim objAlarmBanner As AlarmWndCmdTarget
        Set objAlarmBanner =
        GetSynopticObject.GetSubObject("objAlarmBanner").GetObjectInterface
        objAlarmBanner.HasSpin = Not objAlarmBanner.HasSpin
        Set objAlarmBanner = Nothing
End Sub
```

HelpBtnText, AlarmWndCmdTarget Property

Syntax HelpBtnText = _String

Description This property returns the text which is to appear on the "Help Button". The default text will be used if this field is left blank.

This property is not managed by the "Alarm Banner".

Parameter	Description
None	None

Result String

Example:

```
Sub Click()
        Debug.Print HelpBtnText
End Sub
```

HisLogBackColor, AlarmWndCmdTarget Property

Syntax HisLogBackColor = _Long

Description This property sets or returns the back color of the area containing the alarms history.

This function is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim objWnd As AlarmWndCmdTarget
        Set objWnd = GetSynopticObject.GetSubObject("AlarmWnd").GetObjectInterface
        If objWnd Is Nothing Then Exit Sub
        MsgBox " HisLogBackColor = " & cstr(objWnd.HisLogBackColor
        ),vbInformation,GetProjectTitle
        Set objWnd = Nothing
End Sub
```

HisLogTextColor, AlarmWndCmdTarget Property

Syntax HisLogTextColor = _Long

Description This property sets or returns the text color of the alarm's history.

This function is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim objWnd As AlarmWndCmdTarget
        Set objWnd = GetSynopticObject.GetSubObject("AlarmWnd").GetObjectInterface
        If objWnd Is Nothing Then Exit Sub
        MsgBox " HisLogTextColor = " &
        cstr(objWnd.HisLogTextColor),vbInformation,GetProjectTitle
        Set objWnd = Nothing
End Sub
```

HorizontalSpin, AlarmWndCmdTarget Property

Syntax HorizontalSpin = _Boolean

Description Through this property you set the spin button, used for scrolling active alarms in windows, to show horizontally or vertically. When setting this property with the "True" value, the spin button will show vertically, setting it with the "False" value the button will show horizontally.

This property is not managed by the "Alarm Window" object. Always returns False.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim objAlarmBanner As AlarmWndCmdTarget
        Set objAlarmBanner =
        GetSynopticObject.GetSubObject("objAlarmBanner").GetObjectInterface
        objAlarmBanner.HorizontalSpin = Not objAlarmBanner.HorizontalSpin
        Set objAlarmBanner = Nothing
End Sub
```

IncludeMilliseconds, AlarmWndCmdTarget Property

Syntax IncludeMilliseconds = _Boolean

Description When this property is enabled the milliseconds will also be included in the "Hour" format in the columns of the window which supports this type of data.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Click()
        IncludeMilliseconds = True
        Debug.Print IncludeMilliseconds
End Sub
```

MaxOPCAEEvents, AlarmWndCmdTarget Property

Syntax MaxOPCAEEvents = _Long

Description This property returns the maximum number of events to be displayed in the alarm window.

This property is not supported in Windows CE.(If set, always returns zero)

This property does not supported by the "Alarm Banner" property.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim objAlarmWnd As AlarmWndCmdTarget
        Set objAlarmWnd = GetSynopticObject.GetSubObject("AlarmWnd").GetObjectInterface
        If Not objAlarmWnd Is Nothing Then
                MsgBox "objAlarmWnd's MaxOPCAEEvents are" &
                objAlarmWnd.MaxOPCAEEvents,vbInformation,GetProjectTitle
        Else
                MsgBox "objAlarmWnd is nothing",vbInformation,GetProjectTitle
        End If
End Sub
```

NetworkBackupServerName, AlarmWndCmdTarget Property

Syntax NetworkBackupServerName = _String

Description This property sets or returns the name of any Network Backup Server used for getting alarms to display the window when the primary server, set in the 'NetworkServer' property is in timeout.

Parameter	Description
None	None

Result String

Example:

```
Dim objAlarmWnd As AlarmWndCmdTarget
Public Sub Click()
        Debug.Print objAlarmWnd.NetworkBackupServerName
End Sub
Public Sub SymbolLoading()
        Set objAlarmWnd =
        GetSynopticObject.GetSubObject("AlarmWindow").GetObjectInterface
End Sub
```

NetworkServer, AlarmWndCmdTarget Property

Syntax NetworkServer = _String

Description This property returns the name of the eventual Networking Server from where the alarms are to be retrieved for displaying in the window.

Parameter	Description
None	None

Result String

Example:

```
Sub Click()
        Debug.Print NetworkServer
End Sub
```

OPCAEServer, AlarmWndCmdTarget Property

Syntax OPCAEServer = _String

Description This property returns the name of the eventual OPC AE Server from which alarm notification, to be displayed in the window derives from.

This property is not supported in Windows CE.(If set, always returns an empty string)

Parameter	Description
None	None

Result String

Example:

```
Sub Click()
        Debug.Print OPCAEServer
End Sub
```

RefreshTimePoll, AlarmWndCmdTarget Property

Syntax RefreshTimePoll = _Long

Description This property returns the Polling time of the network. The value is expressed in milliseconds.

Parameter	Description
None	None

Result Long

Example:

```
Sub Click()
        Debug.Print RefreshTimePoll
End Sub
```

ResetAllSelBtnText, AlarmWndCmdTarget Property

Syntax ResetAllSelBtnText = _String

Description This property returns the text which has to appear on the "Reset All Buttons". The default text will be used when this field is left empty.

Parameter	Description
None	None

Result String

Example:
Sub Click()
Debug.Print **ResetAllSelBtnText**
End Sub

ResetSelBtnText, AlarmWndCmdTarget Property

Syntax ResetSelBtnText = _String

Description This property returns the text which has to appear on the "Reset Sel Buttons". When this field is left empty the default text will be used instead.

This property is not support by the "Alarm Banner" object.

Parameter	Description
None	None

Result String

Example:
Sub Click()
Debug.Print **ResetSelBtnText**
End Sub

ScrollTime, AlarmWndCmdTarget Property

Syntax ScrollTime = _Long

Description This property is used for setting the alarm scroll time in milliseconds which will be used when "Autoscroll" property is enabled.

This property is not managed by the "Alarm Window" object. 0 is always returned.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim objAlarmBanner As AlarmWndCmdTarget
        Set objAlarmBanner =
        GetSynopticObject.GetSubObject("objAlarmBanner").GetObjectInterface
        objAlarmBanner.ScrollTime = 1000
        Set objAlarmBanner = Nothing
End Sub
```

ShowAckAllBtn, AlarmWndCmdTarget Property

Syntax ShowAckAlBtn = _Boolean

Description When this property is enabled Movicon will make the command button available for acknowledging all the alarms presented in the Alarm Window without having to select them first.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Click()
        ShowAckAllBtn = True
        Debug.Print ShowAckAllBtn
End Sub
```

ShowAckSelBtn, AlarmWndCmdTarget Property

Syntax ShowAckSelBtn = _Boolean

Description When this property is enabled Movicon will make the command button available for acknowledging the alarms selected in the Alarm Window. In order to carry out this action in RunTime you need to select one or more alarms.

This property is not managed by the "Alarm Banner" object.

Parameter	Description

None	None

Result Boolean

Example:

```
Sub Click()
        ShowAckSelBtn = True
        Debug.Print ShowAckSelBtn
End Sub
```

ShowDateTime, AlarmWndCmdTarget Property

Syntax ShowDateTime = _Boolean

Description This property is used if the event activation date and time must appear in the alarm's text. Setting this property's value to "True" will show the event occurrence date and time in addition to the alarm's text. If set at "False", only the alarm's text will display.

This property is not managed by the "Alarm Window" object. Always returns False.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim objAlarmBanner As AlarmWndCmdTarget
        Set objAlarmBanner =
        GetSynopticObject.GetSubObject("objAlarmBanner").GetObjectInterface
        objAlarmBanner.ShowDateTime = Not objAlarmBanner.ShowDateTime
        Set objAlarmBanner = Nothing
End Sub
```

ShowGetHistoryBtn, AlarmWndCmdTarget Property

Syntax ShowGetHistoryBtn = _Boolean

Description This property shows or hides the "Get History" button.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim objWnd As AlarmWndCmdTarget
        Set objWnd = GetSynopticObject.GetSubObject("AlarmWnd").GetObjectInterface
        If objWnd Is Nothing Then Exit Sub
        objWnd.ShowGetHistoryBtn = Not objWnd.ShowGetHistoryBtn
        Set objWnd = Nothing
End Sub
```

ShowHelpBtn, AlarmWndCmdTarget Property

Syntax ShowHelpBtn = _Boolean

Description When this property is enabled, Movicon will provide the command button for opening a help file for the selected alarm.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Click()
        Dim objAlarmWnd As AlarmWndCmdTarget
        Set objAlarmWnd = GetSynopticObject.GetSubObject("AlarmWnd").GetObjectInterface
        If Not objAlarmWnd Is Nothing Then
                objAlarmWnd.ShowResetAllBtn = Not objAlarmWnd.ShowResetAllBtn
                objAlarmWnd.RecalcLayout
                MsgBox "objAlarmWnd's ShowResetAllBtn is " & objAlarmWnd.ShowResetAllBtn
                ,vbInformation,GetProjectTitle
        Else
                MsgBox "objAlarmWnd is nothing",vbInformation,GetProjectTitle
        End If
End Sub
```

ShowHigherSeverity, AlarmWndCmdTarget Property

Syntax ShowHigherSeverity = _Boolean

Description This property is used for setting the sequence of alarms to show in the Banner according to their severity type. Setting this property to "True", the alarms with the highest severity will be displayed in the banner. In this case the scroll sequence will be based on severity, then activation time and date for those alarms with the same severity starting tieh the most recent to the oldest. Setting this property to "False", the scroll sequence will be based on the alarm activation time, starting with the most recent to the oldest.

This property is not managed by the "Alarm Window" object. Always returns 'False'.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim objAlarmBanner As AlarmWndCmdTarget
        Set objAlarmBanner = GetSynopticObject.GetSubObject("objAlarmBanner").GetObjectInterface
        objAlarmBanner.ShowHigherSeverity = Not objAlarmBanner.ShowHigherSeverity
        Set objAlarmBanner = Nothing
End Sub
```

ShowResetAllBtn, AlarmWndCmdTarget Property

Syntax ShowResetAllBtn = _Boolean

Description when this property is enabled Movicon will make the command button available for resetting all the alarms presented in the Alarm Window without having to select them first providing that these have been silenced with the "Ack Sel" or "Ack All" command.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Click()
        ShowResetAllBtn = True
        Debug.Print ShowResetAllBtn
End Sub
```

ShowResetSelBtn, AlarmWndCmdTarget Property

Syntax ShowResetSelBtn = _Boolean

Description When this property is enabled Movicon will make the command button available for resetting the alarms selected in the Alarm Window. In order to carry out this operation in Runtime you need to select one or more alarms providing that these have been silenced with the "Ack Sel" or "Ack All" command.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Click()
        ShowResetSelBtn = True
        Debug.Print ShowResetSelBtn
End Sub
```

ShowSoundOnBtn, AlarmWndCmdTarget Property

Syntax ShowSoundOnBtn = _Boolean

Description When enabling this property Movicon will make the command button available for activating or deactivating the acoustic sound associated to the unacknowledged alarm priorities.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Click()
        ShowSoundOnBtn = True
        Debug.Print ShowSoundOnBtn
End Sub
```

SpinSize, AlarmWndCmdTarget Property

Syntax SpinSize = _Byte

Description The Alarm Banner's spin button size is set using this property.

The choices are:
0 = small
1 = medium
2 = large

This property is not managed by the "Alarm Window". Always returns 0.

Parameter	Description
None	None

Result Byte

Example:

```
Public Sub Click()
Dim objAlarmBanner As AlarmWndCmdTarget
Set objAlarmBanner = GetSynopticObject.GetSubObject("objAlarmBanner").GetObjectInterface
objAlarmBanner.SpinSize = 2
Set objAlarmBanner = Nothing
End Sub
```

SoundOnBtnText, AlarmWndCmdTarget Property

Syntax SoundOnBtnText = _String

Description This property returns the text which has to appear on the "Sound ON/OFF Buttons". When this field is left empty the default text will be used instead.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result String

Example:

```
Sub Click()
Debug.Print SoundOnBtnText
End Sub
```

SubItemAck, AlarmWndCmdTarget Property

Syntax SubItemAck = _String

Description Permits you to set the text which has to appear as the same of the "Time Ack" column. When this field is left empty the default text will be used instead.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result String

Example:

```
Sub Click()
        Debug.Print SubItemAck
End Sub
```

SubItemAckPos, AlarmWndCmdTarget Property

Syntax SubItemAckPos = _Integer

Description This property sets or returns the position of the "ACK Time" column within the Alarm Window. When setting a new value, the other columns will be automatically re-positioned in the window layout. In addition when setting the "-1", the column will be hidden. The "0" value is used to indicate position of the first column on the left in the window.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result Integer

Example:

```
Sub Click()
        Debug.Print SubItemAckPos
End Sub
```

SubItemAckWidth, AlarmWndCmdTarget Property

Syntax SubItemDurationWidth = _Integer

Description This property indicated the size in pixels of the column within the Alarm Window. When the column is not displayed the value -1 is returned.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result Integer

Example:

```
Sub Click()
        Debug.Print SubItemDurationWidth
End Sub
```

SubItemCondition, AlarmWndCmdTarget Property

Syntax SubItemCondition = _String

Description Permits you to set the text which has to appear as the same of the "Condition" column. When this field is left empty the default text will be used instead.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result String

Example:

```
Sub Click()
        Debug.Print SubItemCondition
End Sub
```

SubItemConditionPos, AlarmWndCmdTarget Property

Syntax SubItemConditionPos = _Integer

Description This property sets or returns the position of the "Condition" column within the Alarm Window. When setting a new value, the other columns will be automatically re-positioned in the window layout. In addition when setting the "-1", the column will be hidden. The "0" value is used to indicate position of the first column on the left in the window.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result Integer

Example:

```
Sub Click()
        Debug.Print SubItemConditionPos
End Sub
```

SubItemConditionWidth, AlarmWndCmdTarget Property

Syntax SubItemConditionWidth = _Integer

Description This property indicates the size in pixels of the column "Condition" inside the window displaying the alarms. The value -1 is returned when the column is not displayed.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result Integer

Example:

```
Sub Click()
        Debug.Print SubItemConditionWidth
End Sub
```

SubItemDuration, AlarmWndCmdTarget Property

Syntax SubItemDuration = _String

Description Permits you to set the text which is to appear as the name of the "Duration" column. When this field is left empty the default text will be used instead.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result String

Example:

```
Sub Click()
        Debug.Print SubItemDuration
End Sub
```

SubItemDurationPos, AlarmWndCmdTarget Property

Syntax SubItemDurationPos = _Integer

Description This property sets or returns the position of the "Duration" column within Alarm Window. When setting a new value, the other columns will be automatically re-positioned in the window layout. In addition when setting the "-1", the column will be hidden. The "0" value is used to indicate position of the first column on the left in the window.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result Integer

Example:

```
Sub Click()
        Debug.Print SubItemDurationPos
End Sub
```

SubItemDurationWidth, AlarmWndCmdTarget Property

Syntax SubItemDurationWidth = _Integer

Description This property indicates the size in pixels of the column within the window displaying the alarms. The value -1 is returned when the column is displayed.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result Integer

Example:

```
Sub Click()
        Debug.Print SubItemDurationWidth
End Sub
```

SubItemImage, AlarmWndCmdTarget Property

Syntax SubItemImage = _String

Description Permits you to set the text which has to appear as the same of the "Image" column. When this field is left empty the default text will be used instead.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result String

Example:

```
Sub Click()
        Debug.Print SubItemImage
End Sub
```

SubItemImagePos, AlarmWndCmdTarget Property

Syntax SubItemImagePos = _Integer

Description This property sets or returns the position of the "Image" column within the Alarm Window. When setting a new value, the other columns will be automatically re-positioned in the window layout. In addition when setting the "-1", the column will be hidden. The "0" value is used to indicate position of the first column on the left in the window.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result Integer

Example:

```
Sub Click()
        Debug.Print SubItemImagePos
End Sub
```

SubItemImageWidth, AlarmWndCmdTarget Property

Syntax SubItemImageWidth = _Integer

Description This property indicates the size in pixels of the column "Image" inside the window displaying the alarms. The value -1 is returned when the column is not displayed.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result Integer

Example:

```
Sub Click()
        Debug.Print SubItemImageWidth
End Sub
```

SubItemOff, AlarmWndCmdTarget Property

Syntax SubItemOff = _String

Description Permits you to set the text which has to appear as the name of the "Time Off" column. When this field is left empty the default text will be used instead.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result String

Example:

```
Sub Click()
        Debug.Print SubItemOff
End Sub
```

SubItemOffPos, AlarmWndCmdTarget Property

Syntax SubItemOffPos = _Integer

Description This property sets or returns the position of the "OFF" column within the Alarm window. When setting a new value, the other columns will be automatically re-positioned in the window layout. In addition when setting the "-1", the column will be hidden. The "0" value is used to indicate position of the first column on the left in the window.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result Integer

Example:

```
Sub Click()
        Debug.Print SubItemOffPos
End Sub
```

SubItemOffWidth, AlarmWndCmdTarget Property

Syntax SubItemOffWidth = _Integer

Description This property indicates the size in pixels of the column inside the window displaying the alarms. The value -1 is returned when the column is not displayed.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result Integer

Example:

```
Sub Click()
        Debug.Print SubItemOffWidth
End Sub
```

SubItemOn, AlarmWndCmdTarget Property

Syntax SubItemOn = _String

Description Permits you set the text to appear as the name of the "Time On" column. When this field is left empty the default text will be used instead.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result String

Example:

```
Sub Click()
        Debug.Print SubItemOn
End Sub
```

SubItemOnPos, AlarmWndCmdTarget Property

Syntax SubItemOnPos = _Integer

Description This property sets or returns the position of the "ON" column within the Alarm window. When setting a new value, the other columns will be automatically re-positioned in the window layout. In addition when setting the "-1", the column will be hidden. The "0" value is used to indicate position of the first column on the left in the window.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result Integer

Example:

```
Sub Click()
        Debug.Print SubItemOnPos
End Sub
```

SubItemOnWidth, AlarmWndCmdTarget Property

Syntax SubItemOnWidth = _Integer

Description This property indicates the size in pixels of the column inside the window displaying the alarms. The value -1 is returned when the column is not displayed.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result Integer

Example:

```
Sub Click()
        Debug.Print SubItemOnWidth
End Sub
```

SubItemReset, AlarmWndCmdTarget Property

Syntax SubItemReset = _String

Description Permits you to set the text to appear as the name for the "Time Reset" column. The default text will be used when this field is left empty.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result String

Example:

```
Sub Click()
        Debug.Print SubItemReset
End Sub
```

SubItemResetPos, AlarmWndCmdTarget Property

Syntax SubItemResetPos = _Integer

Description This property sets or returns the position of the "Reset" column within the Alarm window. When setting a new value, the other columns will be automatically re-positioned in the window layout. In addition when setting the "-1", the column will be hidden. The "0" value is used to indicate position of the first column on the left in the window.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result Integer

Example:

```
Sub Click()
        Debug.Print SubItemResetPos
End Sub
```

SubItemResetWidth, AlarmWndCmdTarget Property

Syntax SubItemResetWidth = _Integer

Description This property indicates the size in pixels of the column inside the window displaying the alarms. The value -1 is returned when the column is not displayed.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result Integer

Example:

```
Sub Click()
        Debug.Print SubItemResetWidth
End Sub
```

SubItemSeverity, AlarmWndCmdTarget Property

Syntax SubItemSeverity = _String

Description Permits you to set the text to appear as the name for the "Severity" column. The default text will be used when this field is left empty.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result String

Example:

```
Sub Click()
        Debug.Print SubItemSeverity
End Sub
```

SubItemSeverityPos, AlarmWndCmdTarget Property

Syntax SubItemSeverityPos = _Integer

Description This property sets or returns the position of the "Severity" column within the Alarm window. When setting a new value, the other columns will be automatically re-positioned in the window layout. In addition when setting the "-1", the column will be hidden. The "0" value is used to indicate position of the first column on the left in the window.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result Integer

Example:

```
Sub Click()
        Debug.Print SubItemSeverityPos
End Sub
```

SubItemSeverityWidth, AlarmWndCmdTarget Property

Syntax SubItemSeverityWidth = _Integer

Description This property indicates the size in pixels of the column inside the window displaying the alarms. The value -1 is returned when the column is not displayed.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result Integer

Example:

```
Sub Click()
        Debug.Print SubItemSeverityWidth
End Sub
```

SubItemStatus, AlarmWndCmdTarget Property

Syntax SubItemStatus = _String

Description Permits you to set the text to appear as the name for the "Status" column. The default text will be used when this field is left empty.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result String

Example:

```
Sub Click()
        Debug.Print SubItemStatus
End Sub
```

SubItemStatusPos, AlarmWndCmdTarget Property

Syntax SubItemStatusPos = _Integer

Description This property sets or returns the position of the "Status" column within the Alarm Manager window. When setting a new value, the other columns will be automatically re-positioned in the window layout. In addition when setting the "-1", the column will be hidden. The "0" value is used to indicate position of the first column on the left in the window.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result Integer

Example:

```
Sub Click()
        Debug.Print SubItemStatusPos
End Sub
```

SubItemStatusWidth, AlarmWndCmdTarget Property

Syntax SubItemStatusWidth = _Integer

Description This property indicates the size in pixels of the column inside the window displaying the alarms. The value -1 is returned when the column is not displayed.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result Integer

Example:

```
Sub Click()
        Debug.Print SubItemStatusWidth
End Sub
```

SubItemText, AlarmWndCmdTarget Property

Syntax SubItemText = _String

Description Permits you to set the text to appear as the name for the "Alarm Description" column. The default text will be used when this field is left empty.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result String

Example:

```
Sub Click()
        Debug.Print SubItemText
End Sub
```

SubItemTextPos, AlarmWndCmdTarget Property

Syntax SubItemTextPos = _Integer

Description This property sets or returns the position of the "Text" column within the Alarm window. When setting a new value, the other columns will be automatically re-positioned in the window layout. In addition when setting the "-1", the column will be hidden. The "0" value is used to indicate position of the first column on the left in the window.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result Integer

Example:

```
Sub Click()
        Debug.Print SubItemTextPos
End Sub
```

SubItemTextWidth, AlarmWndCmdTarget Property

Syntax SubItemTextWidth = _Integer

Description This property indicates the size in pixels of the column inside the window displaying the alarms. The value -1 is returned when the column is not displayed.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result Integer

Example:

```
Sub Click()
        Debug.Print SubItemTextWidth
End Sub
```

SubItemTotalNumAck, AlarmWndCmdTarget Property

Syntax SubItemTotalNumAck = _String

Description Permits you to set the text which has to appear as the name of the "Total Num ACK" column. When this field is left empty the default text will be used instead.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result String

Example:

```
Sub Click()
        Debug.Print SubItemTotalNumAck
End Sub
```

SubItemTotalNumAckPos, AlarmWndCmdTarget Property

Syntax SubItemTotalNumAckPos = _Integer

Description This property sets or returns the position of the "Total Num ACK" column within the Alarm Window. When setting a new value, the other columns will be automatically re-positioned in the window layout. In addition when setting the "-1", the column will be hidden. The "0" value is used to indicate position of the first column on the left in the window.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result Integer

Example:

```
Sub Click()
        Debug.Print SubItemTotalNumAckPos
End Sub
```

SubItemTotalNumAckWidth, AlarmWndCmdTarget Property

Syntax SubItemTotalNumAckWidth = _Integer

Description This property indicates the size in pixels of the column "Total Num ACK" inside the window displaying the alarms. The value -1 is returned when the column is not displayed.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result Integer

Example:

```
Sub Click()
        Debug.Print SubItemTotalNumAckWidth
End Sub
```

SubItemTotalNumOn, AlarmWndCmdTarget Property

Syntax SubItemTotalNumOn = _String

Description Permits you to set the text which has to appear as the same of the "Total Num ON" column. When this field is left empty the default text will be used instead.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result String

Example:

```
Sub Click()
        Debug.Print SubItemTotalNumOn
End Sub
```

SubItemTotalNumOnPos, AlarmWndCmdTarget Property

Syntax SubItemTotalNumOnPos = _Integer

Description This property sets or returns the position of the "Total Num ON" column within the Alarm Window. When setting a new value, the other columns will be automatically re-positioned in the window layout. In addition when setting the "-1", the column will be hidden. The "0" value is used to indicate position of the first column on the left in the window.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result Integer

Example:

```
Sub Click()
        Debug.Print SubItemTotalNumOnPos
End Sub
```

SubItemTotalNumOnWidth, AlarmWndCmdTarget Property

Syntax SubItemTotalNumOnWidth = _Integer

Description This property indicates the size in pixels of the column "Total Num ON" inside the window displaying the alarms. The value -1 is returned when the column is not displayed.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result Integer

Example:

```
Sub Click()
        Debug.Print SubItemTotalNumOnWidth
End Sub
```

SubItemTotalNumReset, AlarmWndCmdTarget Property

Syntax SubItemTotalNumReset = _String

Description Permits you to set the text which has to appear as the same of the "Total Num RESET" column. When this field is left empty the default text will be used instead.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result String

Example:

```
Sub Click()
        Debug.Print SubItemTotalNumReset
End Sub
```

SubItemTotalNumResetPos, AlarmWndCmdTarget Property

Syntax SubItemTotalNumResetPos = _Integer

Description This property sets or returns the position of the "Total Num RESET" column within the Alarm Window. When setting a new value, the other columns will be automatically re-positioned in the window layout. In addition when setting the "-1", the column will be hidden. The "0" value is used to indicate position of the first column on the left in the window.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result Integer

Example:

```
Sub Click()
        Debug.Print SubItemTotalNumResetPos
End Sub
```

SubItemTotalNumResetWidth, AlarmWndCmdTarget Property

Syntax SubItemTotalNumResetWidth = _Integer

Description This property indicates the size in pixels of the column "Total Num RESET" inside the window displaying the alarms. The value -1 is returned when the column is not displayed.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result Integer

Example:

```
Sub Click()
        Debug.Print SubItemTotalNumResetWidth
End Sub
```

SubItemTotalTimeOn, AlarmWndCmdTarget Property

Syntax SubItemTotalTimeOn = _String

Description Permits you to set the text which has to appear as the same of the "Total Time ON" column. When this field is left empty the default text will be used instead.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result String

Example:

```
Sub Click()
        Debug.Print SubItemTotalTimeOn
End Sub
```

SubItemTotalTimeOnPos, AlarmWndCmdTarget Property

Syntax SubItemTotalTimeOnPos = _Integer

Description This property sets or returns the position of the "Total Time ON" column within the Alarm Window. When setting a new value, the other columns will be automatically re-positioned in the window layout. In addition when setting the "-1", the column will be hidden. The "0" value is used to indicate position of the first column on the left in the window.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result Integer

Example:

```
Sub Click()
        Debug.Print SubItemTotalTimeOnPos
End Sub
```

SubItemTotalTimeOnWidth, AlarmWndCmdTarget Property

Syntax SubItemTotalTimeOnWidth = _Integer

Description This property indicates the size in pixels of the column "Total Time ON" inside the window displaying the alarms. The value -1 is returned when the column is not displayed.

This property is not managed by the "Alarm Banner" object.

Parameter	Description
None	None

Result Integer

Example:

```
Sub Click()
        Debug.Print SubItemTotalTimeOnWidth
End Sub
```

7. ButtonCmdTarget

GetCommandsInterfaceOnPressed, ButtonCmdTarget Function

Syntax GetCommandsInterfaceOnPressed()

Description This function gets the CommandsListCmdTarget interface relating to the referenced button's command list. This interface list can be used for modifying the referenced object's "Commands on Pressed" list.

Parameter	Description
None	None

Result Object: returns a CommandsListCmdTarget type object.

Example1:

```
Public Sub Click()
        Dim objButton As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget

        Set objButton =
        GetSynopticObject.GetSubObject("objButton").GetObjectInterface
        Set objCommandList = objButton.GetCommandsInterfaceOnPressed

        Set objCommandList = Nothing
        Set objButton = Nothing
End Sub
```

GetCommandsInterfaceOnRelease, ButtonCmdTarget Function

Syntax GetCommandsInterfaceOnRelease ()

Description This function gets the CommandsListCmdTarget interface relating to the referenced button's command list. This interface can be used for modifying the referenced object's "Commands on Release" list.

Parameter	Description
None	None

Result Object: returns a CommandsListCmdTarget type object.

Example1:

```
Public Sub Click()
        Dim objButton As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget

        Set objButton =
        GetSynopticObject.GetSubObject("objButton").GetObjectInterface
        Set objCommandList = objButton.GetCommandsInterfaceOnRelease

        Set objCommandList = Nothing
        Set objButton = Nothing
End Sub
```

GetCommandsInterfaceWhileDown, ButtonCmdTarget Function

Syntax GetCommandsInterfaceWhileDown()

Description This function gets the CommandsListCmdTarget interface relating to the referenced button's command list. This interface can be used for modifying the referenced object's "Commands While Down" list.

Parameter	Description
None	None

Result Object: returns a CommandsListCmdTarget type object.

Example1:

```
Public Sub Click()
        Dim objButton As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget

        Set objButton =
        GetSynopticObject.GetSubObject("objButton").GetObjectInterface
        Set objCommandList = objButton.GetCommandsInterfaceWhileDown

        Set objCommandList = Nothing
        Set objButton = Nothing
End Sub
```

GetShorcutText, ButtonCmdTarget Function

Syntax GetShorcutText()

Description This function returns the shortcut text set in the referenced button.

Parameter	Description

None	None

Result String

Example1:

```
Public Sub Click()
        Dim objButton As ButtonCmdTarget

        Set objButton =
        GetSynopticObject.GetSubObject("objButton").GetObjectInterface
        MsgBox "Shorcut Text = " &
        objButton.GetShorcutText,vbInformation,GetProjectTitle

        Set objButton = Nothing
End Sub
```

AsciiKeyShortcut, ButtonCmdTarget Property

Syntax AsciiKeyShortcut = _Byte

Description This property sets or returns the ASCII code of the key to be used as accelerator for the referenced button.

Parameter	Description
None	None

Result Byte

Example1:

```
Public Sub Click()
        Dim objButton As ButtonCmdTarget
        Dim bCode As String

        Set objButton =
        GetSynopticObject.GetSubObject("objButton").GetObjectInterface
        bCode = InputBox("Insert Shortcut Key:", "Button
        Prop",Chr(objButton.AsciiKeyShortcut))
        objButton.AsciiKeyShortcut = Asc(bCode)

        Set objButton = Nothing
End Sub
```

Border, ButtonCmdTarget Property

Syntax Border = eBorderType

Description This property sets or returns the border type for the referenced button. Border type can be specified using the eBorderType enumerator or by

inserting the corresponding numeric value:

enum_bt_none (valore 0, Simple)
enum_bt_EDGE_BUMP (valore 1, Bumped)
enum_bt_EDGE_ETCHED (valore 2, etched)
enum_bt_EDGE_RAISED (valore 3, Raised)
enum_bt_EDGE_SUNKEN (valore 4, Sunken)

Parameter	Description
None	None

Result eBorderType

Example1:

```
Public Sub Click()
        Dim objButton As ButtonCmdTarget

        Set objButton =
        GetSynopticObject.GetSubObject("objButton").GetObjectInterface
        objButton.Border = enum_bt_EDGE_RAISED

        Set objButton = Nothing
End Sub
```

ButtonStyle, ButtonCmdTarget Property

Syntax ButtonStyle = eButtonStyle

Description This property sets or returns the referenced button's style type. Style type can be specified using the eButtonStyle enumerator or by inserting the corresponding numeric value:

enum_windowsstylebutton (value 0, Normal)
enum_style3D (value 0, 3D)
enum_yellowlight (value 0,yellow light)
enum_bluelight (value 0, blue light)
enum_greenlight (value green light)
enum_redlight (value 0, red light)
enum_yellowbutton (value 0, button with yellow light)
enum_bluebutton (value 0, button with blue light)
enum_greenbutton (value 0, button with green light)
enum_redbutton (value 0, button with red light)
enum_emergencya (value 0, Emergency A)
enum_emergencyb (valuee 0, Emergency B)
enum_squareb (value 0, Blue square)
enum_squarer (value 0, red square)
enum_squarey (value 0, yellow square)
enum_squareg (value 0, green square)
enum_squaren (value 0, black square)
enum_switcha (value 0, switch A)
enum_switchb (value 0,switch B)
enum_switchc (value 0, switch C)
enum_selectora (value 0, selector A)
enum_selectorb (value 0, seletor B)
enum_selectorc (value 0, selector C)
enum_switch3sthor (value 0, horizontal 3 state swtch)
enum_switch3stver (value 0, horizontal 3 state switch)

enum_selector3sta (value 0, 3 state A selector)
enum_selector3stb (value 0, 3 state B selector)
enum_selector3stc (value 0, 3 state C selector)
enum_selector3std (value 0,3 state D selector)
enum_selector3ste (value 0, 3 state E selector)
enum_selector3stf (value 0, 3 state F selector)
enum_selector3stg (value 0, 3 state G selector)
enum_selector3sth (value 0, central zero 3 state selector)

Parameter	Description
None	None

Result eButtonStyle

Example1:

```
Public Sub Click()
	Dim objButton As ButtonCmdTarget

	Set objButton =
	GetSynopticObject.GetSubObject("objButton").GetObjectInterface
	objButton.ButtonStyle = enum_bluebutton

	Set objButton = Nothing
End Sub
```

Clickable, ButtonCmdTarget Property

Syntax Clickable = _Boolean

Description This property sets or returns the referenced button's "clickable" property. Button will no longer be clickable when this property is set to False.

Parameter	Description
None	None

Result Boolean

Example1:

```
Public Sub Click()
	Dim objButton As ButtonCmdTarget

	Set objButton =
	GetSynopticObject.GetSubObject("objButton").GetObjectInterface
	objButton.Clickable = False

	Set objButton = Nothing
End Sub
```

CommandStateVariable, ButtonCmdTarget Property

Syntax CommandStateVariable = _String

Description This property sets and returns the name of the variable inserted in the referenced button's "Command/State Variable" property. This is the name of the variable that acts on the button's "ON-OFF" command.

Parameter	Description
None	None

Result String

Example1:

```
Public Sub Click()
        Dim objButton As ButtonCmdTarget
        Dim sVarName As String

        GetVariableNameFromList(sVarName)

        Set objButton =
        GetSynopticObject.GetSubObject("objButton").GetObjectInterface
        objButton.CommandStateVariable = sVarName

        Set objButton = Nothing
End Sub
```

CommandType, ButtonCmdTarget Property

Syntax CommandType = eMechanicStyle

Description This property sets or returns the command type for the referenced button. This command type can be specified using the eMechanicStyle enumerator or by inserting the corresponding numeric value:

enum_ms_Command (value 0, Executes Commands)
enum_ms_OnOff (value 1, ON-OFF)
enum_ms_Impulsive (value 2, Impulsive)
enum_ms_TristateHor (value 3, Horizontal Three-States)
enum_ms_TristateVer (value 4,Vertical Three-States)

Parameter	Description
None	None

Result eMechanicStyle

Example1:

```
Public Sub Click()
```

```
Dim objButton As ButtonCmdTarget

Set objButton =
GetSynopticObject.GetSubObject("objButton").GetObjectInterface
objButton.CommandType = enum_ms_OnOff

Set objButton = Nothing
End Sub
```

DisableCommandsOnCheckedState, ButtonCmdTarget Property

Syntax DisableCommandsOnCheckedState = _Boolean

Description This property sets or returns the "Conditioned commands" property value which allows button command lists to execute only when the variable inserted in the "Command/State Variable" is set at zero.

Parameter	Description
None	None

Result Boolean

Example1:

```
Public Sub Click()
        Dim objButton As ButtonCmdTarget
        Set objButton = GetSynopticObject.GetSubObject("Button1").GetObjectInterface
        objButton.DisableCommandsOnCheckedState = True
        Set objButton = Nothing
End Sub
```

EnableShortcut, ButtonCmdTarget Property

Syntax EnableShortcut = _Boolean

Description This property sets or returns the referenced button's "Enable Shortcut" property. Setting this property to False will disable the button's shortcut.

Parameter	Description
None	None

Result Boolean

Example1:

```
Public Sub Click()
        Dim objButton As ButtonCmdTarget
```

```
        Set objButton = GetSynopticObject.GetSubObject("objButton").GetObjectInterface
        objButton.EnableShortcut= True

        Set objButton = Nothing
    End Sub
```

ExecuteCommandsOnMouseMove, ButtonCmdTarget Property

Syntax ExecuteCommandsOnMouseMove = _Boolean

Description This property sets or returns the value of the "Command on Mouse Move" property which consents the button object's "Command on Release" list to execute when mouse key is released even in cases when the mouse cursor moves outside the button area after pressing the button.

Parameter	Description
None	None

Result Boolean

Example1:

```
Public Sub Click()
    Dim objButton As ButtonCmdTarget
    Set objButton = GetSynopticObject.GetSubObject("Button1").GetObjectInterface
    objButton.ExecuteCommandsOnMouseMove = True
    Set objButton = Nothing
End Sub
```

ImageBtnChecked, ButtonCmdTarget Property

Syntax ImageBtnChecked = _String

Description This property sets or returns the name of the image associated to the referenced button's "button checked" status.

Parameter	Description
None	None

Result String

Example1:

```
Public Sub Click()
    Dim objButton As ButtonCmdTarget
```

```
        Dim sVarName As String

        GetVariableNameFromList(sVarName)

        Set objButton =
        GetSynopticObject.GetSubObject("objButton").GetObjectInterface
        objButton.ImageBtnChecked = "ImageChecked.bmp"

        Set objButton = Nothing
End Sub
```

ImageBtnDisabled, ButtonCmdTarget Property

Syntax ImageBtnDisabled = _String

Description This property sets or returns the name of the image associted to the referenced button?s "button disabled" status.

Parameter	Description
None	None

Result String

Example1:

```
Public Sub Click()
        Dim objButton As ButtonCmdTarget
        Dim sVarName As String

        GetVariableNameFromList(sVarName)

        Set objButton =
        GetSynopticObject.GetSubObject("objButton").GetObjectInterface
        objButton.ImageBtnDisabled = "ImageDisabled.bmp"

        Set objButton = Nothing
End Sub
```

ImageBtnPressed, ButtonCmdTarget Property

Syntax ImageBtnPressed = _String

Description This property sets or returns the name of the image associated to the referenced button's "button pressed" status.

Parameter	Description
None	None

Result String

Example1:

```
Public Sub Click()
        Dim objButton As ButtonCmdTarget
        Dim sVarName As String

        GetVariableNameFromList(sVarName)

        Set objButton =
        GetSynopticObject.GetSubObject("objButton").GetObjectInterface
        objButton.ImageBtnPressed = "ImagePressed.bmp"

        Set objButton = Nothing
End Sub
```

ImageBtnReleased, ButtonCmdTarget Property

Syntax ImageBtnReleased = _String

Description This property sets or returns the name of the image associated to the referenced button's "button released" status.

Parameter	Description
None	None

Result String

Example1:

```
        Public Sub Click()
        Dim objButton As ButtonCmdTarget
        Dim sVarName As String

        GetVariableNameFromList(sVarName)

        Set objButton =
        GetSynopticObject.GetSubObject("objButton").GetObjectInterface
        objButton.ImageBtnReleased = "ImageReleased.bmp"

        Set objButton = Nothing
End Sub
```

ImpulsiveTime, ButtonCmdTarget Property

Syntax ImpulsiveTime = _Long

Description This property sets or returns the referenced button's impulsive time. This value is only taken into consideration for certain types of command buttons.

Parameter	Description
None	None

Result Long

Example1:

```
Public Sub Click()
        Dim objButton As ButtonCmdTarget

        Set objButton =
        GetSynopticObject.GetSubObject("objButton").GetObjectInterface
        objButton.ImpulsiveTime = 1000

        Set objButton = Nothing
End Sub
```

OverlapImageText, ButtonCmdTarget Property

Syntax OverlapImageText = _Boolean

Description This property sets or returns the referenced button's "Overlap Image" property. When set at True any image associated to the button will be overlapped with the title text, otherwise image and title will be placed side by side.

Parameter	Description
None	None

Result Boolean

Example1:

```
Public Sub Click()
        Dim objButton As ButtonCmdTarget

        Set objButton =
        GetSynopticObject.GetSubObject("objButton").GetObjectInterface
        objButton.OverlapImageText= True

        Set objButton = Nothing
End Sub
```

RadioBtnNumOptions, ButtonCmdTarget Property

Syntax RadioBtnNumOptions = _Byte

Description This property sets or returns the number of options for the "Option Button" object. This property is ignored if button is not an "Option Button" type.

Parameter	Description
None	None

Result Byte

Example1:

```
Public Sub Click()
        Dim objButton As ButtonCmdTarget
        Dim bCode As String

        Set objButton =
        GetSynopticObject.GetSubObject("objButton").GetObjectInterface
        objButton.RadioBtnNumOptions = 5

        Set objButton = Nothing
End Sub
```

RadioCheckBtnSize, ButtonCmdTarget Property

Syntax RadioCheckBtnSize = eRadioCheckSize

Description This property sets or returns the size for "Option Buttons" or "Check Butoons" objects. This property is ignored if the button is not one of these types. The size can be specified using the eRadioCheckSize enumerator or by inserting the corresponding numeric value:

enum_rcsz_small (value 0, small)
enum_rcsz_medium (value 1, medium)
enum_rcsz_large (valuee 2, big)

Parameter	Description
None	None

Result eButtonSize

Example1:

```
Public Sub Click()
        Dim objButton As ButtonCmdTarget
        Dim bCode As String
```

```
        Set objButton =
        GetSynopticObject.GetSubObject("objButton").GetObjectInterface
        objButton.RadioCheckBtnSize = enum_rcsz_medium

        Set objButton = Nothing
End Sub
```

Round3DStyle, ButtonCmdTarget Property

Syntax Round3DStyle = eBtnRoundLevel

Description This property sets or returns the reference 3D buttons rounded style. The style type can be specified using the eBtnRoundLevel enumerator or by inserting the corresponding numeric values:

enum_brl_none (value 0)
enum_brl_small (value 1)
enum_brl_medium (value 2)
enum_brl_large (value 3)

Parameter	Description
None	None

Result eBtnRoundLevel

Example1:

```
Public Sub Click()
        Dim objButton As ButtonCmdTarget

        Set objButton =
        GetSynopticObject.GetSubObject("objButton").GetObjectInterface
        objButton.Round3DStyle = enum_brl_large

        Set objButton = Nothing
End Sub
```

ShowShortcut, ButtonCmdTarget Property

Syntax ShowShortcut = _Boolean

Description This property sets or returns the referenced button's "Show Shortcut" property. when setting this property to True, any shortcut text associated to the button will be shown next to the objects title text.

Parameter	Description
None	None

Result Boolean

Example1:

```
Public Sub Click()
        Dim objButton As ButtonCmdTarget

        Set                                objButton                                =
        GetSynopticObject.GetSubObject("objButton").GetObjectInterface
        objButton.ShowShortcut= True

        Set objButton = Nothing
End Sub
```

TriStateCentralZero, ButtonCmdTarget Property

Syntax TriStateCentralZero = _Boolean

Description This property sets or returns the referenced button's "Central Zero" property. Setting this property to True will display the button's zero in the central position. This property is only valid for TriSate buttons.

Parameter	Description
None	None

Result Boolean

Example1:

```
Public Sub Click()
        Dim objButton As ButtonCmdTarget

        Set                                objButton                                =
        GetSynopticObject.GetSubObject("objButton").GetObjectInterface
        objButton.TriStateCentralZero= True

        Set objButton = Nothing
End Sub
```

VirtualKeyShortcut, ButtonCmdTarget Property

Syntax VirtualKeyShortcut = eVirtualKey

Description This property sets or returns the function key to be used in combination with the shortcut key defined for the referenced button. The function key type can be specified using the eVirtualKey enumerator or by inserting the corresponding numeric value:

enum_VK_NONE (value 0, No Key)
enum_VK_CTRL (value 1, CTRL Key)
enum_VK_SHFT (value 2, SHIFT Key)
enum_VK_ALT (value 4, ALT Key)

Parameter	Description
None	None

Result eVirtualKey

Example1:

```
Public Sub Click()
        Dim objButton As ButtonCmdTarget

        Set objButton =
        GetSynopticObject.GetSubObject("objButton").GetObjectInterface
        objButton.VirtualKeyShortcut = enum_VK_SHFT

        Set objButton = Nothing
End Sub
```

8. ChartWndCmdTarget

OnErrorRecordset, ChartWndCmdTarget Event

Description Event notified following an error verified while acquiring values used in the chart.

Parameter	Description
RecordsetError As String	Detailed error description.

OnRecordsetMoveNext, ChartWndCmdTarget Event

Description Event occurs during the scrolling of the value set, which complies with the selection query, each time the system acquires a new value.

Parameter	Description
NumRecord As Long	Index of currently pointed record
bRet As Boolean	Enabling while scrolling

OnRecordsetQueryEnd, ChartWndCmdTarget Event

Description Event occurs at the end of acquiring the values which comply with the selection query.

Parameter	Description
None	None

OnRecordsetQueryStart, ChartWndCmdTarget Event

Description Event occurs at the start of the acquiring the values which comply with the selection query.

Parameter	Description
None	None

GetChartInterface, ChartWndCmdTarget Function

Syntax GetChartInterface()

Description This function returns the interface "VtChart" relating to the Chart object inserted in a Screen.

For further information on charts and their configurations, please consult the relevant "First Impression 5.0 On-Line documentation" (VCFI5.HLP) on-line guide found in the Movicon installation folder.

Parameter	Description
None	None

Result Object
If Function has been executed successfully it will retrieve an object of type VtChart if otherwise Nothing is returned.

Example:

```
Option Explicit
Public Sub Click()
        Dim objChart As ChartWndCmdTarget
        Set objChart = GetSynopticObject.GetSubObject("Chart1").GetObjectInterface
        Dim obj As VtChart
        Set obj = objChart.GetChartInterface
        Debug.Print obj.Picture
End Sub
```

LoadExtSettings, ChartWndCmdTarget Function

Syntax LoadExtSettings

Description This function permits the object's relating external file settings to be loaded. This file can be specified in design mode in the "External File settings" property or in the "ExtSettingsFile" interface properties. The extension provided for this file is ".SXML".

Parameter	Description
None	None

Result Boolean

Example:
```
Public Sub Click()
Dim objSymbol As ChartWndCmdTarget
Set objSymbol = GetSynopticObject.GetSubObject("TestObject").GetObjectInterface
If objSymbol Is Nothing Then Exit Sub
objSymbol.ExtSettingsFile = "test.sxml"
objSymbol.LoadExtSettings
Set objSymbol = Nothing
End Sub
```

SaveExtSettings, ChartWndCmdTarget Function

Syntax SaveExtSettings

Description This function permits the objects settings to be save in the relating external settings file. This file can be specified when in design mode in the "Ext. Settings File" property, or using the property from the "ExtSettingsFile" interface. The extension provided for this file is ".SXML".

Parameter	Description
None	None

Result Long

Example:
```
Public Sub Click()
Dim objSymbol As ChartWndCmdTarget
Set objSymbol = GetSynopticObject.GetSubObject("TestObject").GetObjectInterface
If objSymbol Is Nothing Then Exit Sub
objSymbol.ExtSettingsFile = "test.sxml"
objSymbol.SaveExtSettings
Set objSymbol = Nothing
End Sub
```

RecalcLayout, ChartWndCmdTarget Function

Syntax RecalcLayout()

Description This function updates the object's graphical aspect. This function needs to be executed after a change has been made to a property involving the object's graphical aspect such as changing the ElevationVariable.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
        Dim ChartWnd As ChartWndCmdTarget
        Set ChartWnd = GetSynopticObject.GetSubObject("Chart").GetObjectInterface
        If Not ChartWnd Is Nothing Then
                ChartWnd.ElevationVariable = "VAR00003"
                ChartWnd.RecalcLayout
        End If
        Set ChartWnd = Nothing
End Sub
```

AddStackVariable, ChartWndCmdTarget Property

Syntax AddStackVariable = _String

Description This property sets or returns the name of the Push Value Variable'. The Chart currently displays all the values of the assigned array type variable. You can use a 'Push Value' variable in advance for updating the chart's values on this variable's status change.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim ChartWnd As ChartWndCmdTarget
        Set ChartWnd = GetSynopticObject.GetSubObject("Chart").GetObjectInterface
        If Not ChartWnd Is Nothing Then
                Debug.Print ChartWnd.AddStackVariable
        End If
        Set ChartWnd = Nothing
End Sub
```

ArrayType, ChartWndCmdTarget Property

Syntax ArrayType = _Integer

Description This property sets or returns the data array type to be displayed on the chart.

The possible types are:
enum_Byte
enum_Double
enum_Dword
enum_Float
enum_Word

Parameter	Description

None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
	Dim ChartWnd As ChartWndCmdTarget
	Set ChartWnd = GetSynopticObject.GetSubObject("Chart").GetObjectInterface
	If Not ChartWnd Is Nothing Then
		If ChartWnd.ArrayType=enum_Float Then
		ChartWnd.ArrayType=enum_Dword
		Else
		ChartWnd.ArrayType=enum_Float
		End If
		ChartWnd.RecalcLayout
	End If
	Set ChartWnd = Nothing
End Sub
```

BackupLink, ChartWndCmdTarget Property

Syntax BackupLink = _String

Description This property allows you to set the ODBC connection to the associated chart. This property is useful when you need to display data filed in other files.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
	Dim ChartWnd As ChartWndCmdTarget
	Dim sConnectionString As String

	sConnectionString =  "MyProject__BackupLink"  'DSN name

	Set ChartWnd = GetSynopticObject.GetSubObject("Chart").GetObjectInterface
	If Not ChartWnd Is Nothing Then
		ChartWnd.BackupLink = sConnectionString
		ChartWnd.RecalcLayout
	End If
	Set ChartWnd = Nothing
End Sub
```

Border, ChartWndCmdTarget Property

Syntax Border = _Integer

Description This property sets or returns the chart border type.

The possible types are:
enum_bt_EDGE_BUMP
enum_bt_EDGE_ETCHED
enum_bt_EDGE_RAISED
enum_bt_EDGE_SUNKEN
enum_bt_none

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
	Dim ChartWnd As ChartWndCmdTarget
	Set ChartWnd = GetSynopticObject.GetSubObject("Chart").GetObjectInterface
	If Not ChartWnd Is Nothing Then
		If ChartWnd.Border = enum_bt_EDGE_BUMP Then
		ChartWnd.Border = enum_bt_EDGE_RAISED
		Else
		ChartWnd.Border = enum_bt_EDGE_BUMP
		End If
		ChartWnd.RecalcLayout
	End If
	Set ChartWnd = Nothing
End Sub
```

Clickable, ChartWndCmdTarget Property

Syntax Clickable = _Boolean

Description This property is used to define whether the operator can interact with the chart. When this property is disabled, the control will no longer respond when either clicked by the mouse or operated from keyboard.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
	Dim ChartWnd As ChartWndCmdTarget
	Set ChartWnd = GetSynopticObject.GetSubObject("Chart").GetObjectInterface
	If Not ChartWnd Is Nothing Then
```

```
            ChartWnd.Clickable = Not ChartWnd.Clickable
            ChartWnd.RecalcLayout
        End If
        Set ChartWnd = Nothing
End Sub
```

DataDefaultQuery, ChartWndCmdTarget Property

Syntax DataDefaultQuery = _String

Description This property sets or returns the SQL query for data extraction from database (Data Logger) associated to the chart object. Accepts a Sting value.
The "RecalcLayout" function verifies where this property is set with a value. In this case the string is sent to the ODBC and used for fetching data recordsets. The values of the other two "DataFilterBy" and "DataSortBy" properties are used only when the "DataDefaultQuery" has not been set.
This property allows custom SELECT queries to be executed. For example, the following query retrieves a recordset of a maximum of 10,000 values, where each value represents the average of values rescorded within an hour:

DataDefaultQuery = "SELECT TOP 10000 0 As MSecCol, MIN(LocalCol) As LocalCol, AVG(Cosine) As Cosine, AVG(Ramp) As Ramp FROM DataLogger GROUP BY DatePart (dayofyear, LocalCol), DatePart (Hour, LocalCol) ORDER BY LocalCol DESC"

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim objChart As ChartWndCmdTarget
        Set objChart = GetSynopticObject.GetSubObject("Chart1").GetObjectInterface
        Begin Dialog UserDialog 370,154,"ChartWndCmdTarget" ' %GRID:10,7,1,1
        GroupBox 20,7,340,84,"DataDefaultQuery",.GroupBox1
        TextBox 100,28,250,56,.Query,1
        Text 30,28,60,21,"Query",.Text1
        OKButton 20,105,160,42
        CancelButton 190,105,160,42
        End Dialog
        Dim dlg As UserDialog
        dlg.Query = "DELETE FROM Log1sec"
        If Dialog(dlg) <> -1 Then Exit Sub
        objChart.DataDefaultQuery = dlg.Query
        objChart.DataFilterBy = ""
        objChart.DataSortBy = ""
        objChart.RecalcLayout
End Sub
```

DataFilterBy, ChartWndCmdTarget Property

Syntax DataFilterBy = _String

Description This property sets or returns the "Filter" field for extracting data from the database associated to the chart object by using the datalogger. The "Filter" respects the SQL syntax and corresponds to the "WHERE" clause. This clause is only used when accessing the database, this means when the project run is started up. Accepts a String value.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
	Dim objChart As ChartWndCmdTarget
	Set objChart = GetSynopticObject.GetSubObject("Chart1").GetObjectInterface
	Begin Dialog UserDialog 370,154,"ChartWndCmdTarget" ' %GRID:10,7,1,1
	GroupBox 20,7,340,84,"DataFilterBy",.GroupBox1
	TextBox 100,28,250,21,.FromDate
	TextBox 100,56,250,21,.ToDate
	Text 30,28,60,21,"From",.Text1
	Text 30,56,60,21,"To",.Text12
	OKButton 20,105,160,42
	CancelButton 190,105,160,42
	End Dialog
	Dim dlg As UserDialog
	dlg.FromDate = Format(Now, c)
	dlg.ToDate = Format(Now, c)
	If Dialog(dlg) <> -1 Then Exit Sub
	Dim dFromDate As Date
	Dim dToDate As Date
	Dim sDataFilterBy As String
	dFromDate = CDate(dlg.FromDate)
	dToDate = CDate(dlg.ToDate)
	sDataFilterBy = "LocalCol >= { ts '" & Format(dFromDate, "yyyy\-mm\-dd hh\:nn\:ss") & "'
	} AND LocalCol <= { ts '" & Format(dToDate, "yyyy\-mm\-dd hh\:nn\:ss") & "' }"
	objChart.DataDefaultQuery = ""
	objChart.DataFilterBy = sDataFilterBy
	objChart.DataSortBy = "LocalCol DESC"
	objChart.RecalcLayout
End Sub
```

DataSortBy, ChartWndCmdTarget Property

Syntax DataSortBy = _String

Description This property sets or returns the "SortBy" field for extracting data from the database associated to the chart object by means of the datalogger. The "SortBy" field respects the SQL syntax and corresponds to the "ORDER BY" clause. This clause is used only when access is made to the database such as launching the project in run mode. Once the screen is loaded you need to keep in mind that the first record of the chart's values is the most recent in time order. Accepts a String value.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
	Dim objChart As ChartWndCmdTarget
	Set objChart = GetSynopticObject.GetSubObject("Chart1").GetObjectInterface
	Begin Dialog UserDialog 370,154,"ChartWndCmdTarget" ' %GRID:10,7,1,1
	GroupBox 20,7,340,91,"DataSortBy",.GroupBox1
	OptionGroup .Group1
	OptionButton 60,28,280,28,"LocalCol ASC",.OptionButton1
	OptionButton 60,56,280,28,"LocalCol DESC",.OptionButton12
	OKButton 10,105,160,42
	CancelButton 190,105,160,42
	End Dialog
	Dim dlg As UserDialog
	If Dialog(dlg) <> -1 Then Exit Sub
	objChart.DataDefaultQuery = ""
	objChart.DataFilterBy = ""
	Select Case dlg.Group1
	Case 0
	objChart.DataSortBy = "LocalCol ASC"
	Case 1
	objChart.DataSortBy = "LocalCol DESC"
	End Select
	objChart.RecalcLayout
End Sub
```

ElevationVariable, ChartWndCmdTarget Property

Syntax ElevationVariable = _String

Description This property allows you to set the name of the variable which determines the 3D chart's vertical rotation. The 3D charts visual rotation angle is put into effect in runtime by the associated variable.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
	Dim ChartWnd As ChartWndCmdTarget
	Dim tmpString As String
	Set ChartWnd = GetSynopticObject.GetSubObject("Chart").GetObjectInterface
	If Not ChartWnd Is Nothing Then
		tmpString = ChartWnd.RotationVariable
```

```
                ChartWnd.RotationVariable = ChartWnd.ElevationVariable
                ChartWnd.ElevationVariable = tmpString
                ChartWnd.RecalcLayout
        End If
        Set ChartWnd = Nothing
End Sub
```

ExtSettingsFile, ChartWndCmdTarget Property

Syntax ExtSettingsFile = _String

Description This property sets or returns the external configuration file for the referenced object. the file can be alsow specified in design mode in the object's "Configuration File" property. The extension provided for this file is ".SXML".

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
Dim objSymbol As ChartWndCmdTarget
Set objSymbol = GetSynopticObject.GetSubObject("TestObject").GetObjectInterface
If objSymbol Is Nothing Then Exit Sub
objSymbol.ExtSettingsFile = "test.sxml"
objSymbol.SaveExtSettings
Set objSymbol= Nothing
End Sub
```

LinkedDataLogger, ChartWndCmdTarget Property

Syntax LinkedDataLogger = _String

Description This property sets or returns the name of the datalogger linked to the chart.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
```

```
        Dim ChartWnd As ChartWndCmdTarget
        Dim tmpString As String
        Set ChartWnd = GetSynopticObject.GetSubObject("Chart").GetObjectInterface
        If Not ChartWnd Is Nothing Then
                ChartWnd.LinkedDataLogger = "DLR5sec"
                ChartWnd.RecalcLayout
        End If
        Set ChartWnd = Nothing
End Sub
```

NetworkBackupServerName, ChartWndCmdTarget Property

Syntax NetworkBackupServerName = _String

Description This property sets or returns the name of any Network Backup Server used for retrieving data to be displayed in the Chart when the Primary Server, being the one set in the "NetowrkServerName" property,is in timeout.

Parameter	Description
None	None

Result String

Example:

```
Dim objChart As ChartWndCmdTarget
Public Sub Click()
        Debug.Print objChart.NetworkBackupServerName
End Sub
Public Sub SymbolLoading()
        Set objChart =
        GetSynopticObject.GetSubObject("ChartWindow").GetObjectInterface
End Sub
```

NetworkServerName, ChartWndCmdTarget Property

Syntax NetworkServerName = _String

Description This property returns the name of any Network Server where data is to be retrieved for displaying in the Chart.

Parameter	Description
None	None

Result String

Example:

```
Dim objChart As ChartWndCmdTarget
Public Sub Click()
	Debug.Print objChart.NetworkServerName
End Sub
Public Sub SymbolLoading()
	Set objChart =
	GetSynopticObject.GetSubObject("ChartWnd").GetObjectInterface
End Sub
```

NumSamples, ChartWndCmdTarget Property

Syntax NumSamples = _Long

Description This property sets or returns the number of values (samples) which are to be represented in the chart. The default value (20) means that the chart will display 20 values in function with the type of data specified, regardless of the array size which is expressed in bytes and is referred to the variable.

Parameter	Description
None	None

Result Long

Example:

```
Option Explicit
Public Sub Click()
	Dim ChartWnd As ChartWndCmdTarget
	Set ChartWnd = GetSynopticObject.GetSubObject("Chart").GetObjectInterface
	If Not ChartWnd Is Nothing Then
		Dbug.Print ChartWnd.NumSamples
	End If
	Set ChartWnd = Nothing
End Sub
```

RotationVariable, ChartWndCmdTarget Property

Syntax RotationVariable = _String

Description This property allows you to set the name of the variable which determines the "horizontal" rotation of the 3D chart. The 3D chart's rotation angles are visualised in runtime according to the value of the associated variable.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim ChartWnd As ChartWndCmdTarget
        Dim tmpString As String
        Set ChartWnd = GetSynopticObject.GetSubObject("Chart").GetObjectInterface
        If Not ChartWnd Is Nothing Then
                tmpString = ChartWnd.RotationVariable
                ChartWnd.RotationVariable = ChartWnd.ElevationVariable
                ChartWnd.ElevationVariable = tmpString
                ChartWnd.RecalcLayout
        End If
        Set ChartWnd = Nothing
End Sub
```

Title1, ChartWndCmdTarget Property

Syntax Title1 = _String

Description This property allows you to associate a name to the number 1 curve represented in the chart.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim ChartWnd As ChartWndCmdTarget
        Dim tmpString As String
        Set ChartWnd = GetSynopticObject.GetSubObject("Chart").GetObjectInterface
        If Not ChartWnd Is Nothing Then
                ChartWnd.Title1 = "Curve 1"
                ChartWnd.RecalcLayout
        End If
        Set ChartWnd = Nothing
End Sub
```

Title2, ChartWndCmdTarget Property

Syntax Title2 = _String

Description This property allows you to associate a name to the number 2 curve represented in the chart.

Parameter	Description

None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
	Dim ChartWnd As ChartWndCmdTarget
	Dim tmpString As String
	Set ChartWnd = GetSynopticObject.GetSubObject("Chart").GetObjectInterface
	If Not ChartWnd Is Nothing Then
		ChartWnd.Title2 = "Curve 2"
		ChartWnd.RecalcLayout
	End If
	Set ChartWnd = Nothing
End Sub
```

Title3, ChartWndCmdTarget Property

Syntax Title3 = _String

Description This property allows you to associate a name to the number 3 curve represented in the chart.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
	Dim ChartWnd As ChartWndCmdTarget
	Dim tmpString As String
	Set ChartWnd = GetSynopticObject.GetSubObject("Chart").GetObjectInterface
	If Not ChartWnd Is Nothing Then
		ChartWnd.Title3 = "Curve 3"
		ChartWnd.RecalcLayout
	End If
	Set ChartWnd = Nothing
End Sub
```

Title4, ChartWndCmdTarget Property

Syntax Title4 = _String

Description This property allows you to associate a name to the number 4 curve represented in the chart.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim ChartWnd As ChartWndCmdTarget
        Dim tmpString As String
        Set ChartWnd = GetSynopticObject.GetSubObject("Chart").GetObjectInterface
        If Not ChartWnd Is Nothing Then
                ChartWnd.Title4 = "Curve 4"
                ChartWnd.RecalcLayout
        End If
        Set ChartWnd = Nothing
End Sub
```

Title5, ChartWndCmdTarget Property

Syntax Title5 = _String

Description This property allows you to associate a name to the number 5 curve represented in the chart.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim ChartWnd As ChartWndCmdTarget
        Dim tmpString As String
        Set ChartWnd = GetSynopticObject.GetSubObject("Chart").GetObjectInterface
        If Not ChartWnd Is Nothing Then
                ChartWnd.Title5 = "Curve 5"
                ChartWnd.RecalcLayout
        End If
        Set ChartWnd = Nothing
End Sub
```

Title6, ChartWndCmdTarget Property

Syntax Title6 = _String

Description This property allows you to associate a name to the number 6 curve represented in the chart.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
	Dim ChartWnd As ChartWndCmdTarget
	Dim tmpString As String
	Set ChartWnd = GetSynopticObject.GetSubObject("Chart").GetObjectInterface
	If Not ChartWnd Is Nothing Then
		ChartWnd.Title6 = "Curve 6"
		ChartWnd.RecalcLayout
	End If
	Set ChartWnd = Nothing
End Sub
```

Title7, ChartWndCmdTarget Property

Syntax Title7 = _String

Description This property allows you to associate a name to the number 7 curve represented in the chart.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
	Dim ChartWnd As ChartWndCmdTarget
	Dim tmpString As String
	Set ChartWnd = GetSynopticObject.GetSubObject("Chart").GetObjectInterface
	If Not ChartWnd Is Nothing Then
		ChartWnd.Title7 = "Curve 7"
		ChartWnd.RecalcLayout
	End If
	Set ChartWnd = Nothing
End Sub
```

Title8, ChartWndCmdTarget Property

Syntax Title8 = _String

Description This property allows you to associate a name to the number 8 curve represented in the chart.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim ChartWnd As ChartWndCmdTarget
        Dim tmpString As String
        Set ChartWnd = GetSynopticObject.GetSubObject("Chart").GetObjectInterface
        If Not ChartWnd Is Nothing Then
                ChartWnd.Title8 = "Curve 8"
                ChartWnd.RecalcLayout
        End If
        Set ChartWnd = Nothing
End Sub
```

Variable1, ChartWndCmdTarget Property

Syntax Variable1 = _String

Description This property allows you to set the name of the variable associated to the chart's number 1 curve.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim ChartWnd As ChartWndCmdTarget
        Dim tmpString As String
        Set ChartWnd = GetSynopticObject.GetSubObject("Chart").GetObjectInterface
        If Not ChartWnd Is Nothing Then
                ChartWnd.Variable1 = "VAR00001"
                ChartWnd.RecalcLayout
        End If
        Set ChartWnd = Nothing
End Sub
```

Variable2, ChartWndCmdTarget Property

Syntax Variable2 = _String

Description This property allows you to set the name of the variable associated to the chart's number 2 curve.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim ChartWnd As ChartWndCmdTarget
        Dim tmpString As String
        Set ChartWnd = GetSynopticObject.GetSubObject("Chart").GetObjectInterface
        If Not ChartWnd Is Nothing Then
                ChartWnd.Variable2 = "VAR00002"
                ChartWnd.RecalcLayout
        End If
        Set ChartWnd = Nothing
End Sub
```

Variable3, ChartWndCmdTarget Property

Syntax Variable3 = _String

Description This property allows you to set the name of the variable associated to the chart's number 3 curve.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim ChartWnd As ChartWndCmdTarget
        Dim tmpString As String
        Set ChartWnd = GetSynopticObject.GetSubObject("Chart").GetObjectInterface
        If Not ChartWnd Is Nothing Then
                ChartWnd.Variable3 = "VAR00003"
                ChartWnd.RecalcLayout
        End If
        Set ChartWnd = Nothing
End Sub
```

Variable4, ChartWndCmdTarget Property

Syntax Variable4 = _String

Description This property allows you to set the name of the variable associated to the chart's number 4 curve.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim ChartWnd As ChartWndCmdTarget
        Dim tmpString As String
        Set ChartWnd = GetSynopticObject.GetSubObject("Chart").GetObjectInterface
        If Not ChartWnd Is Nothing Then
                ChartWnd.Variable4 = "VAR00004"
                ChartWnd.RecalcLayout
        End If
        Set ChartWnd = Nothing
End Sub
```

Variable5, ChartWndCmdTarget Property

Syntax Variable5 = _String

Description This property allows you to set the name of the variable associated to the chart's number 5 curve.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim ChartWnd As ChartWndCmdTarget
        Dim tmpString As String
        Set ChartWnd = GetSynopticObject.GetSubObject("Chart").GetObjectInterface
        If Not ChartWnd Is Nothing Then
                ChartWnd.Variable5 = "VAR00005"
                ChartWnd.RecalcLayout
        End If
        Set ChartWnd = Nothing
End Sub
```

Variable6, ChartWndCmdTarget Property

Syntax Variable6 = _String

Description This property allows you to set the name of the variable associated to the chart's number 6 curve.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim ChartWnd As ChartWndCmdTarget
        Dim tmpString As String
        Set ChartWnd = GetSynopticObject.GetSubObject("Chart").GetObjectInterface
        If Not ChartWnd Is Nothing Then
                ChartWnd.Variable6 = "VAR00006"
                ChartWnd.RecalcLayout
        End If
        Set ChartWnd = Nothing
End Sub
```

Variable7, ChartWndCmdTarget Property

Syntax Variable7 = _String

Description This property allows you to set the name of the variable associated to the chart's number 7 curve.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim ChartWnd As ChartWndCmdTarget
        Dim tmpString As String
        Set ChartWnd = GetSynopticObject.GetSubObject("Chart").GetObjectInterface
        If Not ChartWnd Is Nothing Then
                ChartWnd.Variable7 = "VAR00007"
                ChartWnd.RecalcLayout
        End If
        Set ChartWnd = Nothing
End Sub
```

Variable8, ChartWndCmdTarget Property

Syntax Variable8 = _String

Description This property allows you to set the name of the variable associated to the chart's number 8 curve.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim ChartWnd As ChartWndCmdTarget
        Dim tmpString As String
        Set ChartWnd = GetSynopticObject.GetSubObject("Chart").GetObjectInterface
        If Not ChartWnd Is Nothing Then
                ChartWnd.Variable8 = "VAR00008"
                ChartWnd.RecalcLayout
        End If
        Set ChartWnd = Nothing
End Sub
```

9. ClientRulesInterface

ClientTimeout, ClientRulesInterface Property

Syntax ClientTimeout = _Long

Description This function sets or returns the timeout (in ms) on the Server's response after a message has been sent by the Client. An error message will be generated when the timeout runs out.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim NetwObj As NetworkClientCmd
        Dim ClientRulesObj As ClientRulesInterface
        Set NetwObj  = GetNetworkClient
        If Not NetwObj Is Nothing Then
                Set ClientRulesObj = NetwObj.GetClientRules("")
                If Not ClientRulesObj Is Nothing Then
                        Debug.Print ClientRulesObj.ClientTimeout
                        Set ClientRulesObj = Nothing
                End If
                Set NetwObj = Nothing
        End If
End Sub
```

DefaultClientUser, ClientRulesInterface Property

Syntax DefaultClientUser = _String

Description This function sets or returns the name of the user with which the Client presents to the Server. This setting has meaning only when the Server project has the "Password Management" enabled. In cases where the user also has to exist in the Server project in order to be acknowledged. By doing this the Client will acquire the rights associated to the user in question, and can get access to variables based on these rights.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim NetwObj As NetworkClientCmd
        Dim ClientRulesObj As ClientRulesInterface
        Set NetwObj  = GetNetworkClient
        If Not NetwObj Is Nothing Then
                Set ClientRulesObj = NetwObj.GetClientRules("")
                If Not ClientRulesObj Is Nothing Then
                        Debug.Print ClientRulesObj.DefaultClientUser
                        Set ClientRulesObj = Nothing
                End If
                Set NetwObj = Nothing
        End If
End Sub
```

Name, ClientRulesInterface Property

Syntax Name = _String

Description This function returns the Server connection to which the rules refer to. When a string is inserted (eg. "ServerRule1'), the IP address must be set, relating to this name, in the 'Server Alias Table' property found in the Client Network settings, otherwise the IP address, of the server to be connected to, can be inserted directly in this field.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim NetwObj As NetworkClientCmd
        Dim ClientRulesObj As ClientRulesInterface
        Set NetwObj  = GetNetworkClient
        If Not NetwObj Is Nothing Then
                Set ClientRulesObj = NetwObj.GetClientRules("")
                If Not ClientRulesObj Is Nothing Then
                        Debug.Print ClientRulesObj.Name
                        Set ClientRulesObj = Nothing
                End If
                Set NetwObj = Nothing
        End If
End Sub
```

PingTime, ClientRulesInterface Property

Syntax PingTime = _Long

Description This function sets or returns the ping time to be used while being connected to the Server (the "0" value voids the ping time usage).

Parameter	Description

None	None

Result Long

Example:

```
Public Sub Click()
        Dim NetwObj As NetworkClientCmd
        Dim ClientRulesObj As ClientRulesInterface
        Set NetwObj  = GetNetworkClient
        If Not NetwObj Is Nothing Then
                Set ClientRulesObj = NetwObj.GetClientRules("")
                If Not ClientRulesObj Is Nothing Then
                        Debug.Print ClientRulesObj.PingTime
                        Set ClientRulesObj = Nothing
                End If
                Set NetwObj = Nothing
        End If
End Sub
```

Priority, ClientRulesInterface Property

Syntax Priority = _Integer

Description This function sets or returns the priority level set for the connection in question. The values are from 0 to 100. The highest number corresponds to the highest priority. Therefore 100 is the highest priority.

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        Dim NetwObj As NetworkClientCmd
        Dim ClientRulesObj As ClientRulesInterface
        Set NetwObj  = GetNetworkClient
        If Not NetwObj Is Nothing Then
                Set ClientRulesObj = NetwObj.GetClientRules("")
                If Not ClientRulesObj Is Nothing Then
                        Debug.Print ClientRulesObj.Priority
                        Set ClientRulesObj = Nothing
                End If
                Set NetwObj = Nothing
        End If
End Sub
```

Protocol, ClientRulesInterface Property

Syntax Protocol = _Integer

Description This function sets or returns the Protocol type which the Client must use for communicating with the Server.

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        Dim NetwObj As NetworkClientCmd
        Dim ClientRulesObj As ClientRulesInterface
        Set NetwObj  = GetNetworkClient
        If Not NetwObj Is Nothing Then
                Set ClientRulesObj = NetwObj.GetClientRules("")
                If Not ClientRulesObj Is Nothing Then
                        Debug.Print ClientRulesObj.Protocol
                        Set ClientRulesObj = Nothing
                End If
                Set NetwObj = Nothing
        End If
End Sub
```

RasStation, ClientRulesInterface Property

Syntax RasStation = _String

Description This function sets or returns the name of the RAS connection (which has to be created beforehand) to executed the connection to the Server by exploiting a telephone line.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim NetwObj As NetworkClientCmd
        Dim ClientRulesObj As ClientRulesInterface
        Set NetwObj  = GetNetworkClient
        If Not NetwObj Is Nothing Then
                Set ClientRulesObj = NetwObj.GetClientRules("")
                If Not ClientRulesObj Is Nothing Then
                        Debug.Print ClientRulesObj.RasStation
                        Set ClientRulesObj = Nothing
                End If
                Set NetwObj = Nothing
        End If
End Sub
```

UseRASStation, ClientRulesInterface Property

Syntax UseRASStation = _Boolean

Description This property allow the use of the RAS station to be enable for connecting to the Server using a telephone line.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim NetwObj As NetworkClientCmd
        Dim ClientRulesObj As ClientRulesInterface
        Set NetwObj  = GetNetworkClient
        If Not NetwObj Is Nothing Then
                Set ClientRulesObj = NetwObj.GetClientRules("")
                If Not ClientRulesObj Is Nothing Then
                        ClientRulesObj.RasStation = "StRAS0001"
                        ClientRulesObj.UseRASStation = True
                        Set ClientRulesObj = Nothing
                End If
                Set NetwObj = Nothing
        End If
End Sub
```

10. CommandAlarmCmdTarget

ConvertPeriodNumToString, CommandAlarmCmdTarget Function

Syntax ConvertPeriodNumToString(_lPeriod)

Description This function converts the eReportPeriod enumerator numeric values in the string requested by the StatisticRptReferencePeriod" function. Period type can be specified using the eReportPeriod enumerator or by inserting the corresponding value:

enum_rp_None (value 0, None)
enum_rp_Today (value 1, Today)
enum_rp_YesterdayorToday (value 2, yesterday and Today)
enum_rp_CurrentWeek (value 3, Current Week)
enum_rp_CurrentMonth (value 4, Current Month)
enum_rp_CurrentYear (value 5, Current Year)
enum_rp_Last7days (value 6, Last 7 Days)
enum_rp_Last30days (value 7, Last 30 Days)
enum_rp_Last60Days (value 8, Last 60 Days)
enum_rp_Last90days (value 9, Last 90 Days)
enum_rp_Last1year (value 10, Last Year)
enum_rp_Last2years (value 11, Last 2 Years)
enum_rp_Last5years (value 12, Last 5 Years)
enum_rp_Last10years (value 13, Last 10 Years)

Parameter	Description
_lPeriod as eReportPeriod	period value to be converted in string.

Result String

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.ConvertPeriodNumToString(enum_rp_Today)

        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
```

```
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.ConvertPeriodNumToString(enum_rp_Today)

        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

GetCommandBaseInterface, CommandAlarmCmdTarget Function

Syntax GetCommandBaseInterface()

Description This function gets the CommandBaseCmdTarget interface relating to the referenced command type.

Parameter	Description
None	None

Result Object: returns a CommandBaseCmdTarget type object.

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget
        Dim objCommandBase As CommandBaseCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        Set objCommandBase = objCommandAlarm.GetCommandBaseInterface

        Set objCommandBase = Nothing
        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = NothingEnd Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget
        Dim objCommandBase As CommandBaseCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
```

```
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        Set objCommandBase = objCommandAlarm.GetCommandBaseInterface

        Set objCommandBase = Nothing
        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

Action, CommandAlarmCmdTarget Property

Syntax Action= eAlarmCommand

Description This property sets or returns the action that executes the referenced Alarm Command. Action type can be specified using the eAlarmCommand 'enumerator or by inserting the corresponding numeric value:

enum_ac_ackall (value 0, Acknowledges All)
enum_ac_rstall (value 1, Resets All)
enum_ac_togglesound (value 2, Enables Sound)
enum_ac_viewreport (value 3, Shows Report)
enum_ac_printreport (value 4, Prints Report)
enum_ac_exportreport (value 5, Exports Report)
enum_ac_ViewTextReport (value 6, Shows Text Report)
enum_ac_PrintTextReport (value 7, Prints Textual Report)
enum_ac_SaveTextReport (value 8, Saves Text Report)
enum_ac_AppendTextReport (value 9, Appends Text Report)
enum_ac_EmbeddedRptView (value 10, Shows Embedded Report)
enum_ac_EmbeddedRptPrint (value11, Prints Embedded Report)
enum_ac_EmbeddedRptSave (value 12, Saves Embedded Report)
enum_ac_EmbeddedRptMail (value 13, Sends Embedded Report)
enum_ac_rststatisticdata (value 14, Reset Statistics)

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result eAlarmCommand

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.Action = enum_ac_ackall
        objCommandList.SaveChanges


        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.Action = enum_ac_ackall
        objCommandList.SaveChanges

        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

AreaFilter,CommandAlarmCmdTarget Property

Syntax AreaFilter= _String

Description This property sets or returns the name of the Alarm Area set in the command. In this way the "Ack All" and "Reset All" commands are executed in the alarms of the area specified.

After having adding or modifying a command on the object's command list you will need to execute the CommandsListCmdTarget interface's SaveChanges method to put changes into effect on the object's command list.
Please be reminded that modifications to command lists remain valid only until the object is unloaded from memory (upon screen closure). When the object is next loaded on screen its command list will be restored with the one associated in design mode. However, its is possible to make command list modifications persistent by associating configuration file to the object and then saving it after having made the modifications and saved the command list.

Parameter	Description
None	None

Result String: Name of Alarms Area

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.AreaFilter = "AREA00001"
        objCommandList.SaveChanges

        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.AreaFilter = "AREA00001"
        objCommandList.SaveChanges

        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

PrintSettingsLandscape, CommandAlarmCmdTarget Property

Syntax PrintSettingsLandscape = _Boolean

Description This property is used for setting the Report page to print horizontally instead of vertically.
This parameter if only considered when a "Print Textual Report", "View Embedded Report", "Save Embedded Report", Print Embedded Report" or "Send Embedded Report" has been selected from the "Action" field. Page will print horizontally when set to "True" and vertically when set to

'False'.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result Boolean

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.PrintSettingsLandscape = True
        objCommandList.SaveChanges

        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.PrintSettingsLandscape = True
        objCommandList.SaveChanges

        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

PrintSettingsPageHeight, CommandAlarmCmdTarget Property

Syntax PrintSettingsPageHeight = _Long

Description This command is used for setting the print page's height. This value is set in millimeters and the -1 value (default value) consents use of the printer's print page height size.
This parameter is only considered if the "Print Embedded Report" command has been selected in the "Action" field.

After having added or modified a command from the object's command list you will need to execute the SaveChanges method from the CommandsListCmdTargetinterface to put changes into effect on the object's command list.
Please be reminded that modifications to command lists only remain valid until the object is unloaded from memory (upon closing screen), after which the command list associated during development mode will be restored when object is next loaded into memory. However, modifications can be made persistent by associating a configuration file to the object then saving it every time modification have been made and saved in the command list.

Parameter	Description
None	None

Result Long

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.PrintSettingsPageHeight = 100
        objCommandList.SaveChanges

        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
```

```
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.PrintSettingsPageHeight = 100
        objCommandList.SaveChanges

        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

PrintSettingsPageWidth, CommandAlarmCmdTarget Property

Syntax PrintSettingsPageWidth = _Long

Description This command is used for setting the print page's width. This value is set in millimeters and the -1 value (default value) consents use of the printer's print page width size.
This parameter is only considered if the "Print Embedded Report" command has been selected in the "Action" field.

After having added or modified a command from the object's command list you will need to execute the SaveChanges method from the CommandsListCmdTargetinterface to put changes into effect on the object's command list.
Please be reminded that modifications to command lists only remain valid until the object is unloaded from memory (upon closing screen), after which the command list associated during development mode will be restored when object is next loaded into memory. However, modifications can be made persistent by associating a configuration file to the object then saving it every time modification have been made and saved in the command list.

Parameter	Description
None	None

Result Long

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.PrintSettingsPageWidth = 70
        objCommandList.SaveChanges

        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
```

```
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.PrintSettingsPageWidth = 70
        objCommandList.SaveChanges

        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

PrintSettingsPortSettings, CommandAlarmCmdTarget Property

Syntax PrintSettingsPortSettings = _String

Description This property sets or returns the print port's configuration string for the referenced Alarm Command. This setting is only used in cases in which a selection has been made from "Network Printer", "Bluetooth Broadcom" or "Bluetooth Microsoft" in the "PrinterPort" property :

File: the name and path of the file which the printer driver is to use for saving print out must be set here (i.e. "\FlashDrv\Output.prn")
Stampante di Rete: the printer network path must be set here (i.e. "\\ServerName\PrinterName")
Bluetooth Broadcom: three values separated by the pipe ('|') character must be entered here. The first value represents the bluetooth card address (i.e. 00:0A:D9:EB:66:C7), the second value represents the service name to be used and the third value represents the channel number.
Bluetooth Microsoft: the bluetooth card address is set here (i.e. 00:0A:D9:EB:66:C7)

After having added or modified a command from the object's command list you must execute the CommandsListCmdTarget interface's SaveChanges method to put modification into effect on object's command list.
Please be reminded that modification to command lists are only valid until the object is unloaded from memory (closing screen), after which the command list associated in design mode will be restored when object is reloaded again. However, command list modifications can be made presistent by associating a configuration file to the object and saving the configuration project after modifying and saving the command list.

Parameter	Description
None	None

Result String

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.PrintSettingsPortSettings = "00:0A:D9:EB:66:C7"
        objCommandList.SaveChanges

        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.PrintSettingsPortSettings = "00:0A:D9:EB:66:C7"
        objCommandList.SaveChanges

        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

PrintSettingsPrinterName, CommandAlarmCmdTarget Property

Syntax PrintSettingsPrinterName = _String

Description This field is used for choosing the printer to sent the report to. The printer can be selected from the PC's local printers. If a printer is not specified in this parameter, the one set for Windows default will be used. The "Printer Choice" option will however priority in this setting.
Cases in which the project has been set for windows CE platform, the list of printers is fixed and shows all those supported by the "PrintCE.dll" tool which are:

- HP PCL 3
- Epson ESC/P 2
- Epson Stylus COLOR
- PocketJet II
- PocketJet 200
- Canon BJ (300 dpi)

- Canon BJ (360 dpi)
- Amtech
- Epson LX (9-pin)
- Adobe PDF file
- MTE W40
- Canon IP90
- Partner M1POS
- SP-T8
- Canon IP100
- Zebra
- MP-300
- O'Neil 4 inch
- O'Neil 3 inch
- HP PCL 5e

This parameter is only considered if the "Print Embedded Report" or "Print Textual Report" command has been selected in the "Action" field.

After having added or modified a command from the object's command list you will need to execute the SaveChanges method from the CommandsListCmdTargetinterface to put changes into effect on the object's command list.
Please be reminded that modifications to command lists only remain valid until the object is unloaded from memory (upon closing screen), after which the command list associated during development mode will be restored when object is next loaded into memory. However, modifications can be made persistent by associating a configuration file to the object then saving it every time modification have been made and saved in the command list.

Parameter	Description
None	None

Result String

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.PrintSettingsPrinterName = "Movicon PDF Writer"
        objCommandList.SaveChanges

        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.PrintSettingsPrinterName = "Movicon PDF Writer"
        objCommandList.SaveChanges

        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

PrintSettingsPrinterPort, CommandAlarmCmdTarget Property

Syntax PrintSettingsPrinterPort = ePrinterPorts

Description This property sets or returns the print port for the referenced Alarm Command. The action type can be specified using the ePrinterPorts enumerator or the corresponding numeric values:

enum_port_Undefined (value -1)
enum_port_Infrared (value 0)
enum_port_COM1 (value 1)
enum_port_COM2 (value 2)
enum_port_COM3 (value 3)
enum_port_COM4 (value 4)
enum_port_COM5 (value 5)
enum_port_COM6 (value 6)
enum_port_COM7 (value 7)
enum_port_COM8 (value 8)
enum_port_File (value 9)
enum_port_NetworkPrinter (value 10)
enum_port_COM9 (value 11)
enum_port_COM10 (value 12)
enum_port_COM11 (value 13)
enum_port_COM12 (value 14)
enum_port_BluetoothBroadcom (value 15)
enum_port_BluetoothMicrosoft (value 16)
enum_port_LPT1 (value 17)
enum_port_USB (value 18)

After having added or modified a command from the object's command list you must execute the CommandsListCmdTarget interface's SaveChanges method to put modification into effect on object's command list.
Please be reminded that modification to command lists are only valid until the object is unloaded from memory (closing screen), after which the command list associated in design mode will be restored when object is reloaded again. However, command list modifications can be made presistent by associating a configuration file to the object and saving the configuration project after modifying and saving the command list.

Parameter	Description

None	None

Result ePrinterPorts

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.PrintSettingsPrinterPort = enum_port_LPT1
        objCommandList.SaveChanges

        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.PrintSettingsPrinterPort = enum_port_LPT1
        objCommandList.SaveChanges

        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

PrintSettingsShowPrintDialog, CommandAlarmCmdTarget Property

Syntax PrintSettingsShowPrintDialog = _Boolean

Description When this option is enabled a dialog window will open before report is printed allowing user to select a printer. The choice of printers will be the ones installed on PC.
This parameter is only considered if the "Print Textual Report" or "Print Embedded Report" commands have been selected from the "Action" field.

After having added or modified a command from the object's command list you will need to execute the SaveChanges method from the CommandsListCmdTargetinterface to put changes into effect on the object's command list.

Please be reminded that modifications to command lists only remain valid until the object is unloaded from memory (upon closing screen), after which the command list associated during development mode will be restored when object is next loaded into memory. However, modifications can be made persistent by associating a configuration file to the object then saving it every time modification have been made and saved in the command list.

Parameter	Description
None	None

Result Boolean

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.PrintSettingsShowPrintDialog = True
        objCommandList.SaveChanges

        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.PrintSettingsShowPrintDialog = True
        objCommandList.SaveChanges

        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

Recipient, CommandAlarmCmdTarget Property

Syntax Recipient = _String

Description The user name of user group to receive email with attached report file is entered in this field.
This parameter is only considered when the "Export and Send Email" or "Send Embedded Report" command has been selected in the "Action field.

After having added or modified a command from the object's command list you will need to execute the SaveChanges method from the CommandsListCmdTargetinterface to put changes into effect on the object's command list.
Please be reminded that modifications to command lists only remain valid until the object is unloaded from memory (upon closing screen), after which the command list associated during development mode will be restored when object is next loaded into memory. However, modifications can be made persistent by associating a configuration file to the object then saving it every time modification have been made and saved in the command list.

Parameter	Description
None	None

Result String

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.Recipient = "Progea"
        objCommandList.SaveChanges

        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
```

```
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.Recipient = "Progea"
        objCommandList.SaveChanges

        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

StatisticRptFile, CommandAlarmCmdTarget Property

Syntax StatisticRptFile = _String

Description This property sets or returns the name of the report file to be used. Options are:

- OrderByDate
- OrderByDuration
- GroupByFrequency
- GroupByThreshold

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result String: name of report to be displayed/printed.

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)
```

```
        objCommandAlarm.StatisticRptFile = "OrderByDate"
        objCommandList.SaveChanges


        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.StatisticRptFile = "OrderByDate"
        objCommandList.SaveChanges


        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

StatisticRptReferenceDate, CommandAlarmCmdTarget Property

Syntax StatisticRptReferenceDate = _String

Description This property sets or returns the reference period to be used for extracting data for displaying/printing in the report. String type value to be inserted must show the start and end date of the desired period, using this format: "dd/mm/yyyy hh:mm:ss dd/mm/yyyy hh:mm:ss".

This property is valid only when the “StatisticRptReferenceDate” property has been set with the “None” value or left empty.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description

None	None

Result String: reference period to be displayed

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.StatisticRptReferenceDate = "01/01/2010 00:00:00
        02/01/2010 23:59:59"
        objCommandList.SaveChanges


        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.StatisticRptReferenceDate = "01/01/2010 00:00:00
        02/01/2010 23:59:59"
        objCommandList.SaveChanges


        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

StatisticRptReferenceDuration, CommandAlarmCmdTarget Property

Syntax StatisticRptReferenceDuration= _String

Description This property sets or returns a filter on the duration of each alarm. The default value is "00:00:00" but a filter can be set for only retrieving alarms from the database with durations lasting longer than a certain time settable in "hh:mm:ss".

After having added or modified a command from the object's command list you must execute the

SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result String: minimum duration for alarms to be displayed/printed.

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.StatisticRptReferenceDuration = "01:30:00"
        objCommandList.SaveChanges


        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.StatisticRptReferenceDuration = "01:30:00"
        objCommandList.SaveChanges


        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

StatisticRptReferencePeriod, CommandAlarmCmdTarget Property

Syntax StatisticRptReferencePeriod = _String

Description This property sets or returns the reference period to be used for extracting day for displaying/printing in the report. String type values allowed are:

- None
- Today
- Yesterday or today
- Current week
- Current month
- Current year
- Last 7 days
- Last 30 days
- Last 60 days
- Last 90 days
- Last 1 years
- Last 2 years
- Last 5 years
- Last 10 years

The "ConvertPeriodNumToString()" function can also be used for converting eReportPeriod value types in the string required by StatisticRptReferenceDate property.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result String: reference period to be displayed.

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget
```

```
        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.StatisticRptReferencePeriod =
        objCommandAlarm.ConvertPeriodNumToString(1)
        objCommandList.SaveChanges


        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.StatisticRptReferencePeriod =
        objCommandAlarm.ConvertPeriodNumToString(1)
        objCommandList.SaveChanges


        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

StatisticRptShowToolbar, CommandAlarmCmdTarget Property

Syntax StatisticRptShowToolbar = _Boolean

Description This property allows the toolbar to be hidden or displayed in the report's preview window. This property is only managed when the report has been created with Crystal Report.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result Boolean

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.StatisticRptShowToolbar = True
        objCommandList.SaveChanges


        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.StatisticRptShowToolbar = True
        objCommandList.SaveChanges


        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

StatisticRptShowTree, CommandAlarmCmdTarget Property

Syntax StatisticRptShowTree = _Boolean

Description This property allows the tree structure to be displayed or hidden in the report's preview window. This property gets managed only when the report has been created with Crystal Report.

After having added or modified a command from the object's command list you must execute the

SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result Boolean

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.StatisticRptShowTree = True
        objCommandList.SaveChanges


        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.StatisticRptShowTree = True
        objCommandList.SaveChanges


        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

TextualRptBottomMargin, CommandAlarmCmdTarget Property

Syntax TextualRptBottomMargin = _Long

Description This property allows the print bottom margin to be set or returned. This margin is set in millimetres and the value -1 (default value) consents to using any default margins retrieved through the driver of the printer being used.
This parameter is only considered when the "Print Textual Report" has been selected in the "Action" field.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result Long

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.TextualRptBottomMargin = 10
        objCommandList.SaveChanges


        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget
```

```
        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.TextualRptBottomMargin = 10
        objCommandList.SaveChanges


        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

TextualRptLeftMargin, CommandAlarmCmdTarget Property

Syntax TextualRptLeftMargin = _Long

Description This property returns or allows you to set the left print margin. This margin must be set in millimeters and the value -1 (default value) consents the use of default print margins recovered from the driver of the printer being used.
This parameter is valid only when the "Print Textual Report" command has been selected in the "Action" field.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result Long

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)
```

```
        objCommandAlarm.TextualRptLeftMargin = 10
        objCommandList.SaveChanges


        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.TextualRptLeftMargin = 10
        objCommandList.SaveChanges


        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

TextualRptMaxPages, CommandAlarmCmdTarget Property

Syntax TextualRptMaxPages = _Long

Description This property returns or allows you to set the maximum number of printed pages with one single "View Textual Report", "Print Textual Report", "Save Textual Report", "Append Textual report" command. The value "0" imposes no limit on the number of pages that can be printed and therefore try not to use this value to avoid occupying too much memory or printer overuse in the event of errors in the data extraction query formulae.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result Long

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.TextualRptMaxPages = 10
        objCommandList.SaveChanges


        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.TextualRptMaxPages = 10
        objCommandList.SaveChanges


        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

TextualRptOutputFile, CommandAlarmCmdTarget Property

Syntax TextualRptOutputFile = _String

Description This property returns or allows you to set the name of the file in which the textual report is to be saved. This parameter is only required by the "Save Textual Report" and Append Textual Report" commands. A file will be created in the project's "DLOGGERS" folder if no file path is specified.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is

downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result String

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.TextualRptOutputFile = "TestReport.rtf"
        objCommandList.SaveChanges


        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.TextualRptOutputFile = "TestReport.rtf"
        objCommandList.SaveChanges


        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

TextualRptRightMargin, CommandAlarmCmdTarget Property

Syntax TextualRptRightMargin = _Long

Description This property gets or allows you to set the right print margin. This margin must be set in millimeters and the value -1 (default value) consents the use of default print margins recovered from the driver of the printer being used.
This parameter is valid only when the "Print Textual Report" command has been selected in the "Action" field.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result Long

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.TextualRptRightMargin = 10
        objCommandList.SaveChanges


        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget
```

```
        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.TextualRptRightMargin = 10
        objCommandList.SaveChanges


        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

TextualRptSQLQuery, CommandAlarmCmdTarget Property

Syntax TextualRptSQLQuery = _String

Description This property allows you to set or return the query to be used for extracting data from a "Textual Report" or "Embedded Report". In the default "Textual Reports" data is extracted from the Historical Log's Alarm Table for default. However you can specify a different table in the query to extract data from (Drivers or SysMsgs).
Query must be in the right context according database used and variable names cannot be inserted to make query dynamic.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result String

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)
```

```
        objCommandAlarm.TextualRptSQLQuery = "Select * From Alarms"
        objCommandList.SaveChanges


        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.TextualRptSQLQuery = "Select * From Alarms"
        objCommandList.SaveChanges


        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

TextualRptTemplateFile, CommandAlarmCmdTarget Property

Syntax TextualRptTemplateFile = _String

Description This property gets or allows you to set the name of the layout file for the textual report that, in addition to the format, also contains some special fields to use as well. File will be searched for in the Project's Resources Folder if path is not specified.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result String

Example1:

```
Public Sub Click()

        Dim objButtonRelease As ButtonCmdTarget

        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.TextualRptTemplateFile = "TemplateRpt.rtf"
        objCommandList.SaveChanges


        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTaDim objCommandAlarm As
        CommandAlarmCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.TextualRptTemplateFile = "TemplateRpt.rtf"
        objCommandList.SaveChanges


        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

TextualRptTopMargin, CommandAlarmCmdTarget Property

Syntax TextualRptTopMargin = _Long

Description This property allows the top print margin to be set or returned. This margin is set in millimetres and the value -1 (default value) consents to using any default margins retrieved through the driver of the printer being used.
This parameter is only considered when the "Print Textual Report" has been selected in the "Action" field.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply

modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result Long

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.TextualRptTopMargin = 10
        objCommandList.SaveChanges


        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandAlarm As CommandAlarmCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandAlarm = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandAlarm.TextualRptTopMargin = 10
        objCommandList.SaveChanges


        Set objCommandAlarm = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

11. CommandBaseCmdTarget

Type, CommandBaseCmdTarget Property

Syntax Type = eCommandTypes

Description This property is read only and returns a value which identifies the referenced command type. The return value is the eCommandTypes enumerator:

enum_ct_synoptic (value 0, Screen Command)
enum_ct_script (value 1, Script Command)
enum_ct_variable (value 2, Variable Command))
enum_ct_changelanguage (value3, Language Command)
enum_ct_menu (value 4, Menù Command)
enum_ct_report (value 5, Report Command)
enum_ct_system (value 6, System Command)
enum_ct_users (value 7, User Command)
enum_ct_help (value 8, Help Command)
enum_ct_alarm (value 9, Alarm Command)
enum_ct_event (value 10, Event Command)

Parameter	Description
None	None

Result eCommandTypes

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandBase As CommandBaseCmdTarget
        Dim objObject As Object

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objObject = objCommandList.GetCommandInterfaceAtPos(0)
        Set objCommandBase = objObject.GetCommandBaseInterface

        MsgBox "Command Type = " &
        CStr(objCommandBase.Type),vbInformation,GetProjectTitle

        Set objObject = Nothing
        Set objCommandBase = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
Dim objRect As DrawCmdTarget
Dim objCommandList As CommandsListCmdTarget
```

```
        Dim objCommandBase As CommandBaseCmdTarget
        Dim objObject As Object

Set objRect = GetSynopticObject.GetSubObject("objRect")
Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objObject = objCommandList.GetCommandInterfaceAtPos(0)
        Set objCommandBase = objObject.GetCommandBaseInterface

MsgBox "Command Type = " &
CStr(objCommandBase.Type),vbInformation,GetProjectTitle

        Set objObject = Nothing
        Set objCommandBase = Nothing
Set objCommandList = Nothing
Set objRect = Nothing
End Sub
```

XmlSettings, CommandBaseCmdTarget Property

Syntax XmlSettings = _string

Description This property is read only and returns the referenced command's xml code.

Parameter	Description
None	None

Result String

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandBase As CommandBaseCmdTarget
        Dim objObject As Object

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objObject = objCommandList.GetCommandInterfaceAtPos(0)
        Set objCommandBase = objObject.GetCommandBaseInterface

        MsgBox "Command XmlSettings = " &
        CStr(objCommandBase.XmlSettings),vbInformation,GetProjectTitle

        Set objObject = Nothing
        Set objCommandBase = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
```

```
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandBase As CommandBaseCmdTarget
        Dim objObject As Object

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objObject = objCommandList.GetCommandInterfaceAtPos(0)
        Set objCommandBase = objObject.GetCommandBaseInterface

        MsgBox "Command XmlSettings = " &
        CStr(objCommandBase.XmlSettings),vbInformation,GetProjectTitle

        Set objObject = Nothing
        Set objCommandBase = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

12. CommandEventCmdTarget

Event, CommandEventCmdTarget Property

Syntax Event = _String

Description This property sets or returns the name of the Event object which the referenced command executes.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result String

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandEvent As CommandEventCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandEvent = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandEvent.Event = "EventTest"
        objCommandList.SaveChanges


        Set objCommandEvent = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
        Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandEvent As CommandEventCmdTarget
```

```
        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandEvent = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandEvent.Event = "EventTest"
        objCommandList.SaveChanges


        Set objCommandEvent = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

GetCommandBaseInterface, CommandEventCmdTarget Function

Syntax GetCommandBaseInterface()

Description This function gets the CommandBaseCmdTarget interface relating to the referenced command type.

Parameter	Description
None	None

Result Object: returns a CommandBaseCmdTarget object type.

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandEvent As CommandEventCmdTarget
        Dim objCommandBase As CommandBaseCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandEvent = objCommandList.GetCommandInterfaceAtPos(0)

        Set objCommandBase = objCommandEvent.GetCommandBaseInterface

        Set objCommandBase = Nothing
        Set objCommandEvent = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandEvent As CommandEventCmdTarget
        Dim objCommandBase As CommandBaseCmdTarget
```

```
        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandEvent = objCommandList.GetCommandInterfaceAtPos(0)

        Set objCommandBase = objCommandEvent.GetCommandBaseInterface

        Set objCommandBase = Nothing
        Set objCommandEvent = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

13. CommandHelpCmdTarget

GetCommandBaseInterface, CommandHelpCmdTarget Function

Syntax GetCommandBaseInterface()

Description This function gets the CommandBaseCmdTarget interface relating to the referenced command type.

Parameter	Description
None	None

Result Object: returns a CommandBaseCmdTarget type object.

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandHelp As CommandHelpCmdTarget
        Dim objCommandBase As CommandBaseCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandHelp = objCommandList.GetCommandInterfaceAtPos(0)

        Set objCommandBase = objCommandHelp.GetCommandBaseInterface

        Set objCommandBase = Nothing
        Set objCommandHelp = Nothing
        Set objCommandList = Nothing
         Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandHelp As CommandHelpCmdTarget
        Dim objCommandBase As CommandBaseCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandHelp = objCommandList.GetCommandInterfaceAtPos(0)

        Set objCommandBase = objCommandHelp.GetCommandBaseInterface

        Set objCommandBase = Nothing
        Set objCommandHelp = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
```

```
End Sub
```

Action, CommandHelpCmdTarget Property

Syntax Action = eHelpMode

Description This proptery sets or returns the action that executes the referenced Help Command. Action type can be specified using the eHelpMode enumerator or by inserting the correspond numeric value:

enum_hm_topic (value 0, Topic)
enum_hm_tooltip (value 1, ToolTip)

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result eHelpMode

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandHelp As CommandHelpCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandHelp = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandHelp.Action = enum_hm_tooltip
        objCommandList.SaveChanges


        Set objCommandHelp = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
```

```
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandHelp As CommandHelpCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandHelp = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandHelp.Action = enum_hm_tooltip
        objCommandList.SaveChanges

        Set objCommandHelp = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

Topic, CommandHelpCmdTarget Property

Syntax Topic= _String

Description This property sets or gets the name of the ToolTip Topic to be displayed according to the "Action" field settings.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result String

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandHelp As CommandHelpCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandHelp = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandHelp.Topic = "Hello!"
        objCommandHelp.SaveChanges
```

```
        Set objCommandHelp = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
  Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandHelp As CommandHelpCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandHelp = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandHelp.Topic = "Hello!"
        objCommandHelp.SaveChanges

        Set objCommandHelp = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

14. CommandLanguageCmdTarget

Language, CommandLanguageCmdTarget Property

Syntax Language= _String

Description This property sets or returns the language which the referenced Language command is to activate. The text represents the name of the column in the String Table.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result String

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandLanguage As CommandLanguageCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandLanguage = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandLanguage.Language = "Italian"
        objCommandList.SaveChanges


        Set objCommandLanguage = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
                Dim objCommandLanguage As CommandLanguageCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
                Set objCommandLanguage =
                objCommandList.GetCommandInterfaceAtPos(0)

                objCommandLanguage.Language = "Italian"
        objCommandList.SaveChanges

                Set objCommandLanguage = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

GetCommandBaseInterface, CommandLanguageCmdTarget Function

Syntax GetCommandBaseInterface()

Description This function gets the CommandBaseCmdTarget interface relating to the referenced command type.

Parameter	Description
None	None

Result Object: returns a CommandBaseCmdTarget type object.

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandLanguage As CommandLanguageCmdTarget
        Dim objCommandBase As CommandBaseCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandLanguage = objCommandList.GetCommandInterfaceAtPos(0)

        Set objCommandBase = objCommandLanguage.GetCommandBaseInterface

        Set objCommandBase = Nothing
        Set objCommandLanguage = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = NothingEnd Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
                Dim objCommandLanguage As CommandLanguageCmdTarget
                Dim objCommandBase As CommandBaseCmdTarget
```

```
        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandLanguage = objCommandList.GetCommandInterfaceAtPos(0)

            Set                    objCommandBase                    =
            objCommandLanguage.GetCommandBaseInterface

        Set objCommandBase = Nothing
            Set objCommandLanguage = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

15. CommandsListCmdTarget

AddToHead, CommandsListCmdTarget Function

Syntax AddToHead(_nCommandType)

Description This method is used for adding a new command at the top of the referenced Command List. The "nCommandType" parameter defines command type to be added. This command type can be specified using the eCommandTypes enumerator or inserting the corresponding numeric value:

enum_ct_synoptic (value 0, Screen Command)
enum_ct_script (value 1, Script Command)
enum_ct_variable (value 2, Variable Command)
enum_ct_changelanguage (value 3, Change Language)
enum_ct_menu (value 4, Menu Command)
enum_ct_report (value 5, Report Command)
enum_ct_system (value 6, System Command)
enum_ct_users (value 7,Users Command)
enum_ct_help (value 8, Help Command)
enum_ct_alarm (value 9, Alarm Command)
enum_ct_event (value 10, Event Command)

Parameter	Description
nCommandType as eCommandTypes	Command type to be added.

Result Object

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease

        objCommandList.AddToHead(enum_ct_variable)

        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
```

```
objCommandList.AddToHead(enum_ct_variable)

Set objCommandList = Nothing
Set objRect = Nothing
End Sub
```

AddToTail, CommandsListCmdTarget Function

Syntax AddToTail(_nCommandType)

Description This method is used for adding a new command at the bottom top of the referenced Command List. The "nCommandType" parameter defines command type to be added. This command type can be specified using the eCommandTypes enumerator or inserting the corresponding numeric value:

enum_ct_synoptic (value 0, Screen Command)
enum_ct_script (value 1, Script Command)
enum_ct_variable (value 2, Variable Command)
enum_ct_changelanguage (value 3, Change Language)
enum_ct_menu (value 4, Menu Command)
enum_ct_report (value 5, Report Command)
enum_ct_system (value 6, System Command)
enum_ct_users (value 7,Users Command)
enum_ct_help (value 8, Help Command)
enum_ct_alarm (value 9, Alarm Command)
enum_ct_event (value 10, Event Command)

Parameter	Description
nCommandType as eCommandTypes	Command type to be added.

Result Object

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList =
        objButtonRelease.GetCommandsInterfaceOnRelease

        objCommandList.AddToTail(enum_ct_variable)

        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
```

```
        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease

        objCommandList.AddToTail(enum_ct_variable)

        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

DiscardChanges, CommandsListCmdTarget Function

Syntax DiscardChanges()

Description This method deletes (unloads) changes made to the command list. Only those changes not saved with the "SavesChanges" command will be unloaded from the command list.

Parameter	Description
None	None

Result None

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList =
        objButtonRelease.GetCommandsInterfaceOnRelease

        objCommandList.DiscardChanges

        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease

        objCommandList.DiscardChanges

        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

GetCommandInterfaceAtPos, CommandsListCmdTarget Function

Syntax GetCommandTypeAtPos(_nIndex)

Description This method lets you know the command type at a certain position on the referenced Command List. The index parameter presents the command's position on the list. The returned numeric value corresponds to the eCommandTypes enumerator:

enum_ct_synoptic (value 0, Screen Command)
enum_ct_script (value 1, Script Command)
enum_ct_variable (value 2,Variable Command)
enum_ct_changelanguage (value 3, Language Command)
enum_ct_menu (value 4, Menu Command)
enum_ct_report (value 5, Report Command)
enum_ct_system (value 6, System Command)
enum_ct_users (value 7, Users Command)
enum_ct_help (value 8, Help Command)
enum_ct_alarm (value 9, Alarm Command)
enum_ct_event (value 10, Event Command)

Parameter	Description
nIndex as Long	Command index of type to get. This is the position in the command list. Index starts from the zero value.

Result eCommandTypes

Example1:

```
Public Sub Click()
	Dim objButtonRelease As ButtonCmdTarget
	Dim objCommandList As CommandsListCmdTarget
	Set objButtonRelease = GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
	Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease

	MsgBox "Command Type = " & objCommandList.GetCommandTypeAtPos(0),vbInformation,GetProjectTitle

	Set objCommandList = Nothing
	Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
	Dim objRect As DrawCmdTarget
	Dim objCommandList As CommandsListCmdTarget

	Set objRect = GetSynopticObject.GetSubObject("objRect")
	Set objCommandList = objRect.GetCommandsInterfaceOnRelease

	MsgBox "Command Type = " & objCommandList.GetCommandTypeAtPos(0),vbInformation,GetProjectTitle

	Set objCommandList = Nothing
	Set objRect = Nothing
```

```
End Sub
```

GetCommandTypeAtPos, CommandsListCmdTarget Function

Syntax GetCommandInterfaceAtPos(_nIndex)

Description This method gets a command object from referenced command list. The index parameter represents the position of the command in the list.

Parameter	Description
nIndex as Long	Command index to be loaded. This index indicates position in the command list. Index starts from the zero value.

Result Object

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objObject As Object

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList =
        objButtonRelease.GetCommandsInterfaceOnRelease

        Set objObject = objCommandList.GetCommandInterfaceAtPos(0)

        Set objObject = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing


        End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objObject As Object

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease

        Set objObject = objCommandList.GetCommandInterfaceAtPos(0)

        Set objObject = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

GetTotNumCommands, CommandsListCmdTarget Function

Syntax GetTotNumCommands

Description This method lets you know how many commands are on the referenced Command List.

Parameter	Description
None	None

Result Long

Example1:

```
Public Sub Click()
    Dim objButtonRelease As ButtonCmdTarget
    Dim objCommandList As CommandsListCmdTarget
    Set objButtonRelease = GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
    Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease

    MsgBox "Number of commands: " & objCommandList.GetTotNumCommands,vbInformation,GetProjectTitle

    Set objCommandList = Nothing
    Set objButtonRelease = Nothing
End Sub
```

Example2:

```
    Public Sub Click()
    Dim objRect As DrawCmdTarget
    Dim objCommandList As CommandsListCmdTarget

    Set objRect = GetSynopticObject.GetSubObject("objRect")
    Set objCommandList = objRect.GetCommandsInterfaceOnRelease

    MsgBox "Number of commands: " & objCommandList.GetTotNumCommands,vbInformation,GetProjectTitle

    Set objCommandList = Nothing
    Set objRect = Nothing
End Sub
```

InsertAfter, CommandsListCmdTarget Function

Syntax InsertAfter(_nIndex, _nCommandType)

Description This method is used for inserting new commands on the referenced Command List. Command will be inserted in the position after the one specified by the nIndex parameter and type specified in the nCommandType parameter. The command type can be specified using the

eCommandTypesenumerator or by inserting the corresponding numeric value:

enum_ct_synoptic (value 0, Screen Command)
enum_ct_script (value 1, Script Command)
enum_ct_variable (value 2, Variable Command)
enum_ct_changelanguage (value 3, Change Language Command)
enum_ct_menu (value 4, Menù Command)
enum_ct_report (value 5, Report Command)
enum_ct_system (value 6, System Command)
enum_ct_users (value 7, Users Command)
enum_ct_help (value 8, Help Command)
enum_ct_alarm (value 9, Alarm Command)
enum_ct_event (valore 10, Event Command)

Parameter	Description
nCommandType as eCommandTypes	Command type to be added on list.
nIndex as Long	Command index after which new one will be inserted. This is the position on the command list. This index starts with from the zero value.

Result Object

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList =
        objButtonRelease.GetCommandsInterfaceOnRelease
        objCommandList.InsertAfter(0, enum_ct_variable)
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        objCommandList.InsertAfter(0, enum_ct_variable)
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

InsertBefore, CommandsListCmdTarget Function

Syntax InsertBefore (_nIndex, _nCommandType)

Description This method is used for inserting new commands on the referenced Command List. Command will be inserted in the position before the one

specified by the nIndex parameter and type specified in the nCommandType parameter. The command type can be specified using the eCommandTypes enumerator or by inserting the corresponding numeric value:

enum_ct_synoptic (value 0, Screen Command)
enum_ct_script (value 1, Script Command)
enum_ct_variable (value 2, Variable Command)
enum_ct_changelanguage (value 3, Change Language Command)
enum_ct_menu (value 4, Menù Command)
enum_ct_report (value 5, Report Command)
enum_ct_system (value 6, System Command)
enum_ct_users (value 7, Users Command)
enum_ct_help (value 8, Help Command)
enum_ct_alarm (value 9, Alarm Command)
enum_ct_event (valore 10, Event Command)

Parameter	Description
nCommandType as eCommandTypes	Command type to be inserted on list.
nIndex as Long	Command index before which the new one is to be inserted. This is the position on the command list. Index starts with the zero value.

Result Long

Example1:

```
Public Sub Click()
    Dim objButtonRelease As ButtonCmdTarget
    Dim objCommandList As CommandsListCmdTarget
    Set objButtonRelease =
    GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
    Set objCommandList =
    objButtonRelease.GetCommandsInterfaceOnRelease
    objCommandList.InsertBefore(0, enum_ct_variable)
    Set objCommandList = Nothing
    Set objButtonRelease = Nothing
End Sub
```

Example2:

```
    Public Sub Click()
    Dim objRect As DrawCmdTarget
    Dim objCommandList As CommandsListCmdTarget
    Set objRect = GetSynopticObject.GetSubObject("objRect")
    Set objCommandList = objRect.GetCommandsInterfaceOnRelease
    objCommandList.InsertBefore(0, enum_ct_variable)
    Set objCommandList = Nothing
    Set objRect = Nothing
End Sub
```

MoveToHead, CommandsListCmdTarget Function

Syntax MoveToHead(_nIndex)

Description This method allows the command specified in the nIndex parameter to move to the top of the referenced Command List in first position.

Parameter	Description
nIndex as Long	Command index to be moved at the top of the list. This is the position in the command list. Index starts with the zero value.

Result Boolean

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList =
        objButtonRelease.GetCommandsInterfaceOnRelease
        objCommandList.MoveToHead(2)
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        objCommandList.MoveToHead(2)
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

MoveToTail, CommandsListCmdTarget Function

Syntax MoveToTail(_nIndex)

Description This method allows the command specified by the nIndex parameter to move to the bottom of the referenced Command List.

Parameter	Description
nIndex as Long	Index of the command which is to be moved to the end of the list. This is the positin in the command list. Index starts with the value zero.

Result Boolean

Example1:

```
Public Sub Click()
```

```
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Set objButtonRelease = GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        objCommandList.MoveToTail(1)
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
 End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        objCommandList.MoveToTail(1)
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

RemoveAll, CommandsListCmdTarget Function

Syntax RemoveAll

Description This method allows all the commands from the referenced command list to be removed.

Parameter	Description
None	None

Result None

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Set objButtonRelease = GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        objCommandList.RemoveAll()
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
```

```
        objCommandList.RemoveAll()
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

RemoveAtPos, CommandsListCmdTarget Function

Syntax RemoveAtPos(_nIndex)

Description This method allows the command specified by the nIndex parameter to be removed from the referenced Command List.

Parameter	Description
nIndex as Long	Index of the command to be removed. This is the position in the command list. Index starts from the zero value.

Result Boolean

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList =
        objButtonRelease.GetCommandsInterfaceOnRelease
        objCommandList.RemoveAtPos(1)
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        objCommandList.RemoveAtPos(1)
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

RemoveFromHead, CommandsListCmdTarget Function

Syntax RemoveFromHead()

Description This method allows the first command to be removed from the referenced Command List.

Parameter	Description
None	None

Result Boolean

Example1:

```
Public Sub Click()
	Dim objButtonRelease As ButtonCmdTarget
	Dim objCommandList As CommandsListCmdTarget
	Set objButtonRelease =
	GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
	Set objCommandList =
	objButtonRelease.GetCommandsInterfaceOnRelease
	objCommandList.RemoveFromHead()
	Set objCommandList = Nothing
	Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
	Dim objRect As DrawCmdTarget
	Dim objCommandList As CommandsListCmdTarget
	Set objRect = GetSynopticObject.GetSubObject("objRect")
	Set objCommandList = objRect.GetCommandsInterfaceOnRelease
	objCommandList.RemoveFromHead()
	Set objCommandList = Nothing
	Set objRect = Nothing
End Sub
```

RemoveFromTail, CommandsListCmdTarget Function

Syntax RemoveFromTail()

Description This method allows the last command to be removed from the referenced Command List.

Parameter	Description
None	None

Result Boolean

Example1:

```
Public Sub Click()
	Dim objButtonRelease As ButtonCmdTarget
	Dim objCommandList As CommandsListCmdTarget
	Set objButtonRelease =
	GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
```

```
        Set objCommandList =
        objButtonRelease.GetCommandsInterfaceOnRelease
        objCommandList.RemoveFromTail()
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        objCommandList.RemoveFromTail()
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

SaveChanges, CommandsListCmdTarget Function

Syntax SaveChanges()

Description This method allows you to save the referenced Command List for the object in question. Any modifications to the command list will only be put into effect and executable by the object after the SaveChange method has been invoked.

Caution: Modifications to the Object command lists, except for Button objects, will be applied only if when the object's Command List already contains at least one command at its initialization. In cases where the object in question has been associated with a configuration file, the save command of this configuration file will also save any new Command Lists and in this case the next time page is loaded the new command list will be apply when control is initialized.

Parameter	Description
None	None

Result Boolean

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList =
        objButtonRelease.GetCommandsInterfaceOnRelease
        objCommandList.SaveChanges()
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        objCommandList.SaveChanges()
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

SetAtPos, CommandsListCmdTarget Function

Syntax SetAtPos(_nIndex, _nCommandType)

Description This method allows you to modify the command type identified by the nIndex parameter in the referenced Command List. Existing commands will be reset with the new type specified in the nCommandType parameter. This function cannot add new commands but can only reset those already existing. Command type can be specified using the eCommandTypes enumerator or by using the corresponding numeric value:

enum_ct_synoptic (value 0, Screen Command)
enum_ct_script (value 1, Script Command)
enum_ct_variable (value 2, Variable Command)
enum_ct_changelanguage (value 3, Language Command)
enum_ct_menu (value 4, Menu Command)
enum_ct_report (value 5, Report Command)
enum_ct_system (value 6, System Command)
enum_ct_users (value 7, User Command)
enum_ct_help (value 8, Help Command)
enum_ct_alarm (value 9, Alarm Command)
enum_ct_event (value 10, Event Command)

Parameter	Description
nCommandType as eCommandTypes	Command type to set.
nIndex as Long	Command index that will be set with new type. This is the position on the command list. This index starts with from the zero value.

Result Object

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        objCommandList.SetAtPos(0, enum_ct_variable)
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
```

```
        Dim objCommandList As CommandsListCmdTarget
        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        objCommandList.SetAtPos(0, enum_ct_variable)
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

SwapCommands, CommandsListCmdTarget Function

Syntax SwapCommands(_nIndex1, _nIndex2)

Description This method allows two commands in the referenced command list to swap over places.

Parameter	Description
nIndex2 as Long	Index of second command to be swapped over. This is the position in the command list. Index starts from the zero value.
nIndex1 as Long	Index of first command to be swapped over. This is the position in the command list. Index starts from the zero value.

Result Boolean

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList =
        objButtonRelease.GetCommandsInterfaceOnRelease
        objCommandList.SwapCommands(1,3)
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        objCommandList.SwapCommands(1,3)
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

16. CommandMenuCmdTarget

Menu, CommandMenuCmdTarget Property

Syntax Menu = _String

Description This property sets or returns the name of the Menu which is to be activated by the references Menu Command.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result String

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandMenu As CommandMenuCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandMenu = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandMenu.Menu = "MenuTest"
        objCommandList.SaveChanges


        Set objCommandMenu = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
```

```
        Dim objCommandMenu As CommandMenuCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandMenu = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandMenu.Menu = "MenuTest"
        objCommandList.SaveChanges


        Set objCommandMenu = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

XPos, CommandMenuCmdTarget Property

Syntax XPos = _Long

Description This property sets or returns the left horizontal position of the Menu window to be opened using the referenced Menu Command. This value is expressed in pixels ("0" value for mouse position).

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result Long

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget

        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandMenu As CommandMenuCmdTarget

        Set objButtonRelease =
GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandMenu = objCommandList.GetCommandInterfaceAtPos(0)
```

```
        objCommandMenu.XPos = 100
        objCommandList.SaveChanges


        Set objCommandMenu = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandMenu As CommandMenuCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandMenu = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandMenu.XPos = 100
        objCommandList.SaveChanges


        Set objCommandMenu = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

YPos, CommandMenuCmdTarget Property

Syntax YPos = _Long

Description This property sets or returns the top vertical position of the Menu window to be opened using the referenced Menu Command. This value is expressed in pixels ("-1" value for mouse position).

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result Long

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget

        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandMenu As CommandMenuCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandMenu = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandMenu.YPos = 100
        objCommandList.SaveChanges


        Set objCommandMenu = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandMenu As CommandMenuCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandMenu = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandMenu.YPos = 100
        objCommandList.SaveChanges


        Set objCommandMenu = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

GetCommandBaseInterface, CommandMenuCmdTarget Function

Syntax GetCommandBaseInterface()

Description This function gets the CommandBaseCmdTarget interface relating to the referenced command type.

Parameter	Description
None	None

Result Object: returns a CommandBaseCmdTarget object type.

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandMenu As CommandMenuCmdTarget
        Dim objCommandBase As CommandBaseCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandMenu = objCommandList.GetCommandInterfaceAtPos(0)

        Set objCommandBase = objCommandMenu.GetCommandBaseInterface

        Set objCommandBase = Nothing
        Set objCommandMenu = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
        End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandMenu As CommandMenuCmdTarget
        Dim objCommandBase As CommandBaseCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandMenu = objCommandList.GetCommandInterfaceAtPos(0)

        Set objCommandBase = objCommandMenu.GetCommandBaseInterface

        Set objCommandBase = Nothing
        Set objCommandMenu = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

17. CommandReportCmdTarget

GetCommandBaseInterface, CommandReportCmdTarget Function

Syntax GetCommandBaseInterface()

Description This function gets the CommandBaseCmdTarget interface relating to the referenced command type.

Parameter	Description
None	None

Result Object: returns a CommandBaseCmdTarget type object.

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget
        Dim objCommandBase As CommandBaseCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        Set objCommandBase = objCommandReport.GetCommandBaseInterface

        Set objCommandBase = Nothing
        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget
        Dim objCommandBase As CommandBaseCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        Set objCommandBase = objCommandReport.GetCommandBaseInterface

        Set objCommandBase = Nothing
        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
```

End Sub

Action, CommandReportCmdTarget Property

Syntax Action= eReportMode

Description This property sets or returns the action that must execute the referenced Report/Recipe Command. This type of action can be specified using the eReportMode enumerator or inserting the corresponding numeric value:

enum_rm_Show (value 0, Shows Synchro)
enum_rm_Print (value 1, Prints Syncho)
enum_rm_ShowSafe (value 2, Shows Report)
enum_rm_PrintSafe (value 3, Prints Report)
enum_rm_MoveFirst (value 4, Moves to First)
enum_rm_MoveLast (value 5, Moves to Last)
enum_rm_MovePrev (value 6, Moves to previous)
enum_rm_MoveNext (value 7, Moves to Next)
enum_rm_Activate (value 8, Activates)
enum_rm_Save (value 9, Saves)
enum_rm_Delete (value 10, Deletes)
enum_rm_Requery (value 11, Query Filter)
enum_rm_ExecuteQuery (value 12, Executes Query)
enum_rm_ExportSafe (value 13, Exports Report)
enum_rm_DataAnalisys (value 14, Data Analysis)
enum_rm_ViewTextReport (value 15,Shows Text Report)
enum_rm_PrintTextReport (value 16, Prints Text Report)
enum_rm_SaveTextReport (value 17, Saves Text Report)
enum_rm_AppendTextReport (value 18, Appends Text Report)
enum_rm_ExportRecipe (value 19, Exports Recipe)
enum_rm_ImportRecipe (value 20, Imports Recipe)
enum_rm_ExportAndSendMail (value 21, Exports and Sends Email)
enum_rm_EmbeddedRptView (value 22, Shows Embedded Report)
enum_rm_EmbeddedRptPrint (value 23, Prints Embedded Report)
enum_rm_EmbeddedRptSave (value 24, Saves Embedded Report)
enum_rm_EmbeddedRptMail (value 25, Sends Embedded Report)
enum_rm_ReadRecipe (value 26, Read)

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result eReportMode

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.Action = enum_rm_Show
        objCommandList.SaveChanges


        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.Action = enum_rm_Show
        objCommandList.SaveChanges

        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

DLR, CommandReportCmdTarget Property

Syntax DLR = _String

Description This property allows you to read or set the name of the DataLogger or Recipe for which the referenced Report/Recipe command is to be executed.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made

persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result String

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.DLR = “DataLogger1”
        objCommandList.SaveChanges

        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.DLR = “DataLogger1”
        objCommandList.SaveChanges

        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

EmbeddedReportName, CommandReportCmdTarget Property

Syntax EmbeddedReportName = _String

Description This property is used for reading or writing the name of the Movicon "Report" for which the command, selected from the "Action" field is to be executed. This property will only enable when a Movicon "Embedded

Report" is being used.

Ater having added or modified a command from the object's command list you will need to execute the SaveChanges method from the CommandsListCmdTarget interface to put the object's command list modifications into effect.
Please also remember that modifications to command lists remain valid only until the object is unload from memory (screen closure), after which the command list associated in development mode will be restored when object is loaded into memory again. However, command list modifications can be made persistent by associating a configuration file to the object and saving it after having modified and saved the command list.

Parameter	Description
None	None

Result String

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.EmbeddedReportName = "Report1"
        objCommandList.SaveChanges

        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As CommandReportCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.EmbeddedReportName = "Report1"
        objCommandList.SaveChanges

        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

Height, CommandReportCmdTarget Property

Syntax Height = _Long

Description This property sets or returns the Report's preview window height. Value is expressed in pixels ("0'" value is used for default size). This parameter is only accepted if report has been created with Crystal Report.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result Long

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.Height = 400
        objCommandList.SaveChanges

        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)
```

```
        objCommandReport.Height = 400
        objCommandList.SaveChanges

        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

Landscape, CommandReportCmdTarget Property

Syntax Landscape = _Boolean

Description This property is used for setting the Report page with a vertical or horizontal landscape.
This parameter will only be acknowledged if the "Print Textaul Report", "View Embedded Report", "Save Embedded Report", "Print Embedded Report" or "Send Embedded Report" command has been selected in the "Action" field. Setting this property to "False", the report page will assume a vertical landscape. Setting this property to "True"page will assume a horizontal landscape.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result Boolean

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.Landscape = True
        objCommandList.SaveChanges
```

```
        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.Landscape = True
        objCommandList.SaveChanges

        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

PageHeight, CommandReportCmdTarget Property

Syntax PageHeight = _Long

Description This command is used for setting the print page's height. This value is set in millimeters and the -1 value (default value) consents use of the printer's print page height size.
This parameter is only considered if the "Print Embedded Report" command has been selected in the "Action" field.

After having added or modified a command from the object's command list you will need to execute the SaveChanges method from the CommandsListCmdTargetinterface to put changes into effect on the object's command list.
Please be reminded that modifications to command lists only remain valid until the object is unloaded from memory (upon closing screen), after which the command list associated during development mode will be restored when object is next loaded into memory. However, modifications can be made persistent by associating a configuration file to the object then saving it every time modification have been made and saved in the command list.

Parameter	Description
None	None

Result Long

Example1:

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.PageHeight = 400
        objCommandList.SaveChanges

        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.PageHeight= 400
        objCommandList.SaveChanges

        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

PageWidth, CommandReportCmdTarget Property

Syntax PageWidth = _Long

Description This command is used for setting the print page's width. This value is set in millimeters and the -1 value (default value) consents use of the printer's print page width size.
This parameter is only considered if the "Print Embedded Report" command has been selected in the "Action" field.

After having added or modified a command from the object's command list you will need to execute the SaveChanges method from the CommandsListCmdTargetinterface to put changes into effect on the object's command list.
Please be reminded that modifications to command lists only remain valid until the object is unloaded from memory (upon closing screen), after which the command list associated during development mode will be restored when object is next loaded into memory. However, modifications can be made persistent by associating a configuration file to the object then saving

it every time modification have been made and saved in the command list.

Parameter	Description
None	None

Result Long

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.PageWidth= 200
        objCommandList.SaveChanges

        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.PageWidth= 200
        objCommandList.SaveChanges

        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

PortSettings, CommandReportCmdTarget Property

Syntax PortSettings = _String

Description This property sets or returns the print port's configuration string for the referenced Report/Recipe Command. This setting is only used in cases in which a selection has been made from "Network Printer", "Bluetooth Broadcom" or "Bluetooth Microsoft" in the "PrinterPort" property :

File: the name and path of the file which the printer driver is to use for saving

print out must be set here (i.e. "\FlashDrv\Output.prn")
Stampante di Rete: the printer network path must be set here (i.e. "\\ServerName\PrinterName")
Bluetooth Broadcom: three values separated by the pipe ('|') character must be entered here. The first value represents the bluetooth card address (i.e. 00:0A:D9:EB:66:C7), the second value represents the service name to be used and the third value represents the channel number.

Bluetooth Microsoft: the bluetooth card address is set here (i.e. 00:0A:D9:EB:66:C7)

After having added or modified a command from the object's command list you must execute the CommandsListCmdTarget interface's SaveChanges method to put modification into effect on object's command list.
Please be reminded that modification to command lists are only valid until the object is unloaded from memory (closing screen), after which the command list associated in design mode will be restored when object is reloaded again. However, command list modifications can be made presistent by associating a configuration file to the object and saving the configuration project after modifying and saving the command list.

Parameter	Description
None	None

Result String

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.PortSettings = "00:0A:D9:EB:66:C7"
        objCommandList.SaveChanges

        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.PortSettings = "00:0A:D9:EB:66:C7"
        objCommandList.SaveChanges
```

```
        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

PrinterName, CommandReportCmdTarget Property

Syntax PrinterName = _String

Description This field is used for choosing the printer to sent the report to. The printer can be selected from the PC's local printers. If a printer is not specified in this parameter, the one set for Windows default will be used. The "Show Print Dialog" option will however priority in this setting.
Cases in which the project has been set for windows CE platform, the list of printers is fixed and shows all those supported by the "PrintCE.dll" tool which are:

- HP PCL 3
- Epson ESC/P 2
- Epson Stylus COLOR
- PocketJet II
- PocketJet 200
- Canon BJ (300 dpi)
- Canon BJ (360 dpi)
- Amtech
- Epson LX (9-pin)
- Adobe PDF file
- MTE W40
- Canon IP90
- Partner M1POS
- SP-T8
- Canon IP100
- Zebra
- MP-300
- O'Neil 4 inch
- O'Neil 3 inch
- HP PCL 5e

This parameter is only considered if the "Print Embedded Report" or "Print Textual Report" command has been selected in the "Action" field.

After having added or modified a command from the object's command list you will need to execute the SaveChanges method from the CommandsListCmdTargetinterface to put changes into effect on the object's command list.
Please be reminded that modifications to command lists only remain valid until the object is unloaded from memory (upon closing screen), after which the command list associated during development mode will be restored when object is next loaded into memory. However, modifications can be made persistent by associating a configuration file to the object then saving it every time modification have been made and saved in the command list.

Parameter	Description
None	None

Result String

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.PrinterName = "Movicon PDF Writer"
        objCommandList.SaveChanges

        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.PrinterName = "Movicon PDF Writer"
        objCommandList.SaveChanges

        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

PrinterPort, CommandReportCmdTarget Property

Syntax PrinterPort = ePrinterPorts

Description This property sets or returns the print port for the referenced Report/Recipe Command. The action type can be specified using the ePrinterPorts enumerator or the corresponding numeric values:

enum_port_Undefined (value -1)
enum_port_Infrared (value 0)
enum_port_COM1 (value 1)
enum_port_COM2 (value 2)
enum_port_COM3 (value 3)

enum_port_COM4 (value 4)
enum_port_COM5 (value 5)
enum_port_COM6 (value 6)
enum_port_COM7 (value 7)
enum_port_COM8 (value 8)
enum_port_File (value 9)
enum_port_NetworkPrinter (value 10)
enum_port_COM9 (value 11)
enum_port_COM10 (value 12)
enum_port_COM11 (value 13)
enum_port_COM12 (value 14)
enum_port_BluetoothBroadcom (value 15)
enum_port_BluetoothMicrosoft (value 16)
enum_port_LPT1 (value 17)
enum_port_USB (value 18)

After having added or modified a command from the object's command list you must execute the CommandsListCmdTarget interface's SaveChanges method to put modification into effect on object's command list.
Please be reminded that modification to command lists are only valid until the object is unloaded from memory (closing screen), after which the command list associated in design mode will be restored when object is reloaded again. However, command list modifications can be made presistent by associating a configuration file to the object and saving the configuration project after modifying and saving the command list.

Parameter	Description
None	None

Result ePrinterPorts

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.PrinterPort = enum_port_LPT1
        objCommandList.SaveChanges

        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget
```

```
        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.PrinterPort = enum_port_LPT1
        objCommandList.SaveChanges

        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

RecipeCSVSeparator, CommandReportCmdTarget Property

Syntax RecipeCSVSeparator= _Integer

Description This property sets or returns the separator for the ".csv" file when using the "Import Recipe" and "Export Recipe" commands. The character for default is ";". This property's numeric value corresponds to the character's decimal value, for instance 59 stands for the ";" character.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result Integer

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.RecipeCSVSeparator = 59
```

```
        objCommandList.SaveChanges

        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.RecipeCSVSeparator = 59
        objCommandList.SaveChanges

        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

Recipient,CommandReportCmdTarget Property

Syntax Recipient = _String

Description This property is used for setting or returning the recipent user name or group user name to send emails with attached report files created with "Export and send mail" or "Send Embedded Report" command.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result String

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.Recipient = "Administrator"
        objCommandList.SaveChanges

        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.Recipient = "Administrator"
        objCommandList.SaveChanges

        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

ReportExportFormat, CommandReportCmdTarget Property

Syntax ReportExportFormat = eReportExportFormat

Description This property sets or returns the format of the file where report is to be exported using the "Export Report" command. Format type can be specified using the eReportExportFormat enumerator or by inserting the corresponding numeric value:

enum_re_Pdf (value 0, Pdf)
enum_re_Html (value 1, Html)
enum_re_Txt (value 2, Txt)
enum_re_Csv (value 3, Csv)
enum_re_Xls (value 4, Xls)
enum_re_Mht (value 5, Mht)
enum_re_Rtf (value 6, Rtf)
enum_re_Jpeg (value 7, Jpeg)

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.

Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result eReportExportFormat

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.ReportExportFormat = enum_re_Csv
        objCommandList.SaveChanges

        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.ReportExportFormat = enum_re_Csv
        objCommandList.SaveChanges

        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

ReportReferencePeriod, CommandReportCmdTarget Property

Syntax ReportReferencePeriod = eReportPeriod

Description This property sets or returns the reference period used for extracting data to be displayed/printed in report. The period value can be specified using the eReportPeriod enumerator or by inserting the corresponding numeric value:

enum_rp_None (value 0, None)
enum_rp_Today (value 1,Today)
enum_rp_YesterdayorToday (value 2, Yesterday and Today)
enum_rp_CurrentWeek (value 3, Current Week)
enum_rp_CurrentMonth (value 4, Current Month)
enum_rp_CurrentYear (value 5,Current Year)
enum_rp_Last7days (value 6, Last 7 Days)
enum_rp_Last30days (value 7, Last 30 Days)
enum_rp_Last60Days (value 8,Last 60 Days)
enum_rp_Last90days (value 9, Last 90 Days)
enum_rp_Last1year (value 10, Last Year)
enum_rp_Last2years (value 11, Last 2 Years)
enum_rp_Last5years (value 12, Last 5 Years)
enum_rp_Last10years (value 13, Last 10 Years)

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result eReportPeriod

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.ReportReferencePeriod = enum_rp_Today
        objCommandList.SaveChanges
```

```
        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.ReportReferencePeriod = enum_rp_Today
        objCommandList.SaveChanges

        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

ReportShowFilterByDate, CommandReportCmdTarget Property

Syntax ReportShowFilterByDate= _Boolean

Description When set at True, this property will display a dialog window when the Report opens enabling the user to insert the date and time for query to filter the desired data to be shown in the Report.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result Boolean

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.ReportShowFilterByDate = True
        objCommandList.SaveChanges

        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.ReportShowFilterByDate = True
        objCommandList.SaveChanges

        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

ReportShowToolbar, CommandReportCmdTarget Property

Syntax ReportShowToolbar = _Boolean

Description This property allows the toolbar to be hidden or shown in the report preview window. This property is only managed when report is created with Crystal Report.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result Boolean

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.ReportShowToolbar = True
        objCommandList.SaveChanges

        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.ReportShowToolbar = True
        objCommandList.SaveChanges


        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

ReportShowTree, CommandReportCmdTarget Property

Syntax ReportShowTree = _Boolean

Description This property allows you to display or hide the tree structure in the report's preview window. This property will be ignored if report is not created using Crystal Report.

After having added or modified a command from the object's command list you must execute the

SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result Boolean

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.ReportShowTree = True
        objCommandList.SaveChanges


        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.ReportShowTree = True
        objCommandList.SaveChanges

        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

ShowPrintDialog, CommandReportCmdTarget Property

Syntax ShowPrintDialog = _Boolean

Description When this option is enabled, a dialog window will open for setting the setting of the printer to be used before printing the report. Therefore it will be possible to choose which printer to use among those available in the PC.
This parameter is only considered when the "Print Textual Report" or "Print Embedded Report"command has been selected from the "Action" field.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result Boolean

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.ShowPrintDialog = True
        objCommandList.SaveChanges

        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget
```

```
        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.ShowPrintDialog = True
        objCommandList.SaveChanges

        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

TextualRptBottomMargin, CommandReportCmdTarget Property

Syntax TextualRptBottomMargin = _Long

Description This property allows the print bottom margin to be set or returned. This margin is set in millimetres and the value -1 (default value) consents to using any default margins retrieved through the driver of the printer being used.
This parameter is only considered when the "Print Textual Report" has been selected in the "Action" field.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result Long

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)
```

```
        objCommandReport.TextualRptBottomMargin = 10
        objCommandList.SaveChanges


        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)
         objCommandReport.TextualRptBottomMargin = 10
        objCommandList.SaveChanges


        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

TextualRptLeftMargin, CommandReportCmdTarget Property

Syntax TextualRptLeftMargin = _Long

Description This property returns or allows you to set the left print margin. This margin must be set in millimeters and the value -1 (default value) consents the use of default print margins recovered from the driver of the printer being used.
This parameter is valid only when the "Print Textual Report" command has been selected in the "Action" field.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description

None	None

Result Long

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.TextualRptLeftMargin = 10
        objCommandList.SaveChanges


        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.TextualRptLeftMargin = 10
        objCommandList.SaveChanges


        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

TextualRptMaxPages, CommandReportCmdTarget Property

Syntax TextualRptMaxPages = _Long

Description This property returns or allows you to set the maximum number of printed pages with one single "View Textual Report", "Print Textual Report", "Save Textual Report", "Append Textual report" command. The value "0" imposes no limit on the number of pages that can be printed and therefore try not to use this value to avoid occupying too much memory or printer overuse in the event of errors in the data extraction query formulae.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result Long

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.TextualRptMaxPages = 10
        objCommandList.SaveChanges


        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.TextualRptMaxPages = 10
        objCommandList.SaveChanges


        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

TextualRptOutputFile, CommandReportCmdTarget Property

Syntax TextualRptOutputFile = _String

Description This property returns or allows you to set the name of the file in which the textual report is to be saved. This parameter is only required by the "Save Textual Report" and Append Textual Report" commands. A file will be created in the project's "DLOGGERS" folder if no file path is specified.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result String

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.TextualRptOutputFile = "TestReport.rtf"
        objCommandList.SaveChanges


         Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
```

```
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.TextualRptOutputFile = "TestReport.rtf"
        objCommandList.SaveChanges


        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

TextualRptRightMargin, CommandReportCmdTarget Property

Syntax TextualRptRightMargin = _Long

Description This property gets or allows you to set the right print margin. This margin must be set in millimeters and the value -1 (default value) consents the use of default print margins recovered from the driver of the printer being used.
This parameter is valid only when the "Print Textual Report" command has been selected in the "Action" field.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result Long

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
```

```
        Dim objCommandReport As CommandReportCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.TextualRptRightMargin = 10
        objCommandList.SaveChanges


        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.TextualRptRightMargin = 10
        objCommandList.SaveChanges


        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

TextualRptSQLQuery, CommandReportCmdTarget Property

Syntax TextualRptSQLQuery = _String

Description This property is used for setting or returning the query to be used for extracting data from the DataLogger or Recipe selected from a "Textual Report" or "Embedded Report".
Query must be in the right context according database used and variable names cannot be inserted to make query dynamic.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result String

Example1:

```
Public Sub Click()

        Dim objButtonRelease As ButtonCmdTarget

        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.TextualRptSQLQuery = "Select * From DataLogger1"
        objCommandList.SaveChanges


        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.TextualRptSQLQuery = "Select * From DataLogger1"
        objCommandList.SaveChanges

        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

TextualRptTemplateFile, CommandReportCmdTarget Property

Syntax TextualRptTemplateFile = _String

Description This property gets or allows you to set the name of the layout file for the textual report that, in addition to the format, also contains some special fields to use as well. File will be searched for in the Project's Resources Folder if path is not specified.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result String

Example1:

```
Public Sub Click()

        Dim objButtonRelease As ButtonCmdTarget

        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.TextualRptTemplateFile = "TemplateRpt.rtf"
        objCommandList.SaveChanges


        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.TextualRptTemplateFile = "TemplateRpt.rtf"
        objCommandList.SaveChanges


        Set objCommandReport = Nothing
```

```
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

TextualRptTopMargin, CommandReportCmdTarget Property

Syntax TextualRptTopMargin = _Long

Description This property allows the top print margin to be set or returned. This margin is set in millimetres and the value -1 (default value) consents to using any default margins retrieved through the driver of the printer being used.
This parameter is only considered when the "Print Textual Report" has been selected in the "Action" field.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result Long

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReportAs CommandReportCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandReport= objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.TextualRptTopMargin = 10

        Set objCommandReport= Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReportAs CommandReportCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandReport= objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.TextualRptTopMargin = 10

        Set objCommandReport= Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

XPos, CommandReportCmdTarget Property

Syntax XPos = _Long

Description This property sets or returns the left horizontal position of the Report preview window. This value is expressed in pixels ("0" value used for default position). This parameter is only valid for reports created with Crystal Report.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result Long

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget
```

```
        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.XPos = 50
        objCommandList.SaveChanges


        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
                Dim objCommandReport As CommandReportCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.XPos = 50
        objCommandList.SaveChanges


        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

YPos, CommandReportCmdTarget Property

Syntax YPos = _Long

Description This property sets or returns the top vertical position of the Report preview window. This value is expressed in pixels ("-1" value used for default position). This parameter is only valid for reports created with Crystal Report.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description

None	None

Result Long

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.YPos = 50
        objCommandList.SaveChanges



        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.YPos = 50
        objCommandList.SaveChanges


        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

18. CommandScriptCmdTarget

GetCommandBaseInterface, CommandScriptCmdTarget Function

Syntax GetCommandBaseInterface()

Description This function gets the CommandBaseCmdTarget interface relating to the referenced command type.

Parameter	Description
None	None

Result Object: returns a CommandBaseCmdTarget object type.

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandScript As CommandScriptCmdTarget
        Dim objCommandBase As CommandBaseCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandScript = objCommandList.GetCommandInterfaceAtPos(0)

        Set objCommandBase = objCommandScript.GetCommandBaseInterface

        Set objCommandBase = Nothing
        Set objCommandScript = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandScript As CommandScriptCmdTarget
        Dim objCommandBase As CommandBaseCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandScript = objCommandList.GetCommandInterfaceAtPos(0)

        Set objCommandBase = objCommandScript.GetCommandBaseInterface

        Set objCommandBase = Nothing
        Set objCommandScript = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
```

```
End Sub
```

Action, CommandScriptCmdTarget Property

Syntax Action= eOpenSynopticMode

Description This property sets or returns the action that executes the referenced Script Command. Action type can be specified using the eRunScriptMode enumerator or by inserting the corresponding numeric value:

enum_ops_runnormal (value 0, Runs at normal)
enum_ops_runandwait (value 1, Runs synchronized)
enum_ops_runsafe (value 2, Runs in a separate process (Safe Mode))
enum_ops_stop (value 3, Stops)
enum_ops_unload (value 4,unloads)

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result eOpenSynopticMode

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandScript As CommandScriptCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandScript = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandScript.Action = enum_ops_runnormal
        objCommandList.SaveChanges


        Set objCommandScript = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
```

```
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandScript As CommandScriptCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandScript = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandScript.Action = enum_ops_runnormal
        objCommandList.SaveChanges


        Set objCommandScript = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

MoreInstanceAllowed, CommandScriptCmdTarget Property

Syntax MoreInstanceAllowed= _Boolean

Description When set to True, this property allows more instances to be executed at the same time from the same Basic Script. In order for this to go into effect a number major to the number one must be entered in the Basic Script's "Maxi. Instances" property.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result Boolean

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
```

```
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandScript As CommandScriptCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandScript = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandScript.MoreInstanceAllowed = False
        objCommandList.SaveChanges


        Set objCommandScript = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandScript As CommandScriptCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandScript = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandScript.MoreInstanceAllowed = False
        objCommandList.SaveChanges


        Set objCommandScript = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

Parameters, CommandScriptCmdTarget Property

Syntax Parameters = _String

Description This property sets or returns the list of parameters to be passed to the script when called by the referenced Script Command. These parameter are always read as strings and must be separated by the comma (",").

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result String

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandScript As CommandScriptCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandScript = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandScript.Parameters = "Param1,Param2"
        objCommandList.SaveChanges

        Set objCommandScript = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandScript As CommandScriptCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandScript = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandScript.Parameters = "Param1,Param2"
        objCommandList.SaveChanges

        Set objCommandScript = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

Script, CommandScriptCmdTarget Property

Syntax Script = _String

Description This property sets or returns the name of the Basic Script to be executed or aborted by the referenced Script Command.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to

command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result Integer

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandScript As CommandScriptCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandScript = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandScript.Script = "Script1"
        objCommandList.SaveChanges

        Set objCommandScript = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandScript As CommandScriptCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandScript = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandScript.Script = "Script1"
        objCommandList.SaveChanges

        Set objCommandScript = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

SynchroTimeout, CommandScriptCmdTarget Property

Syntax SynchroTimeout= _Long

Description This property sets or returns the maximum timeout to be considered when Basic Script is executed in synchro mode by the referenced Script Command.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result Long

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandScript As CommandScriptCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandScript = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandScript.SynchroTimeout = 1000
        objCommandList.SaveChanges


        Set objCommandScript = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandScript As CommandScriptCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
```

```
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandScript = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandScript.SynchroTimeout = 1000
        objCommandList.SaveChanges


        Set objCommandScript = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

SynopticName, CommandSynopticCmdTarget Property

Syntax SynopticName = _String

Description This property sets or returns the name of the Screen to be opened using the referenced Screen Command.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result String

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSynoptic As CommandSynopticCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList =
        objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandSynoptic =
        objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSynoptic.SynopticName = “Screen1”
        objCommandList.SaveChanges
```

```
        Set objCommandSynoptic = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSynoptic As CommandSynopticCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandSynoptic =
        objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSynoptic.SynopticName = “Screen1”
        objCommandList.SaveChanges


        Set objCommandSynoptic = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

Width, CommandReportCmdTarget Property

Syntax Width = _Long

Description This property sets or returns the width of the Report's preview window. This value is expressed in pixels (with value "0" for default sizes). This parameter is only valid for reports created with Crystal Report.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result Long

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.Width = 600
        objCommandList.SaveChanges

        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandReport As CommandReportCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandReport = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandReport.Width = 600
        objCommandList.SaveChanges


        Set objCommandReport = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

19. CommandSynopticCmdTarget

GetCommandBaseInterface, CommandSynopticCmdTarget Function

Syntax GetCommandBaseInterface()

Description This function gets the CommandBaseCmdTarget interface relating to the referenced command type.

Parameter	Description
None	None

Result Object: returns a CommandBaseCmdTarget object type.

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSynoptic As CommandSynopticCmdTarget
        Dim objCommandBase As CommandBaseCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandSynoptic = objCommandList.GetCommandInterfaceAtPos(0)

        Set objCommandBase = objCommandSynoptic.GetCommandBaseInterface

        Set objCommandBase = Nothing
        Set objCommandSynoptic = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSynoptic As CommandSynopticCmdTarget
        Dim objCommandBase As CommandBaseCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandSynoptic = objCommandList.GetCommandInterfaceAtPos(0)

        Set objCommandBase = objCommandSynoptic.GetCommandBaseInterface

        Set objCommandBase = Nothing
        Set objCommandSynoptic = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

Action, CommandSynopticCmdTarget Property

Syntax Action= eRunScriptMode

Description This property sets or returns the action that must execute the referenced Screen command. The action type can be specified using the eOpenSynopticMode enumerator or by inserting the corresponding numeric value:

enum_ops_opennormal (value 0, Opens as Normal (change page))
enum_ops_openmodal (value 1, Opens as Modal (pop-up))
enum_ops_openframe (value 2, Opens as Frame (multi-monitor))
enum_ops_opensafe (value 3, Opens in a separate process (Safe Mode))
enum_ops_print (value 4, Prints)
enum_ops_close (value 5, Closes and returns to previous)
enum_ops_executesynapse (value 6, executes Synapses)
enum_ops_opennext (value 7, Opens Nest (next Change page ID))
enum_ops_openprev (value 8,Opens Previous (previous change page ID))
enum_ops_captureprint (value 9, Captures and prints)

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result eRunScriptMode

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSynoptic As CommandSynopticCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandSynoptic = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSynoptic.Action = enum_ops_opennormal
```

```
        objCommandList.SaveChanges


        Set objCommandSynoptic = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSynoptic As CommandSynopticCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandSynoptic = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSynoptic.Action = enum_ops_opennormal
        objCommandList.SaveChanges


        Set objCommandSynoptic = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

Height, CommandSynopticCmdTarget Property

Syntax Height = _Long

Description This property sets or returns the height of the window opened with the referenced Screed Command. This value is expressed in pixels (value '0' is used as default size). This parameter is only taken into consideration when the command used is either "Open Modal" or "Open Frame".

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result Long

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSynoptic As CommandSynopticCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandSynoptic = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSynoptic.Height = 400
        objCommandList.SaveChanges

        Set objCommandSynoptic = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSynoptic As CommandSynopticCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandSynoptic = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSynoptic.Height = 400
        objCommandList.SaveChanges

        Set objCommandSynoptic = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

KeeproportionsOnPrint, CommandSynopticCmdTarget Property

Syntax KeeproportionsOnPrint = _Boolean

Description When set at True, this property keeps the same proportions on print as seen on screen. Otherwise, when disabled (left at default) both height and width will be adapted to the size of the whole page printed on. This parameter goes into effect for both the "Script" and "Capture and Print" commands.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object

will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result **Boolean**

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSynoptic As CommandSynopticCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandSynoptic = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSynoptic.KeeproportionsOnPrint = False
        objCommandList.SaveChanges

        Set objCommandSynoptic = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSynoptic As CommandSynopticCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandSynoptic = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSynoptic.KeeproportionsOnPrint = False
        objCommandList.SaveChanges

        Set objCommandSynoptic = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

Monitor, CommandSynopticCmdTarget Property

Syntax Monitor = _Long

Description This property sets or returns the number of the Monitor on which the Screen is to be opened with the referenced Screen Command. This parameter is only taken into consideration if the "Open Frame" command is used.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result **Boolean**

Example1:

```
Public Sub Click()
Dim objButtonRelease As ButtonCmdTarget
Dim objCommandList As CommandsListCmdTarget
Dim objCommandSynoptic As CommandSynopticCmdTarget

Set objButtonRelease =
GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
Set objCommandSynoptic = objCommandList.GetCommandInterfaceAtPos(0)

objCommandSynoptic. Monitor = 2
objCommandList.SaveChanges

Set objCommandSynoptic = Nothing
Set objCommandList = Nothing
Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
Dim objRect As DrawCmdTarget
Dim objCommandList As CommandsListCmdTarget
Dim objCommandSynoptic As CommandSynopticCmdTarget

Set objRect = GetSynopticObject.GetSubObject("objRect")
Set objCommandList = objRect.GetCommandsInterfaceOnRelease
Set objCommandSynoptic = objCommandList.GetCommandInterfaceAtPos(0)

objCommandSynoptic. Monitor = 2
objCommandList.SaveChanges

Set objCommandSynoptic = Nothing
Set objCommandList = Nothing
Set objRect = Nothing
```

```
End Sub
```

ParameterFile, CommandSynopticCmdTarget Property

Syntax ParameterFile = _String

Description This property sets or returns the parameter file's name and any path when opening screen passing parameters.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result Integer

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSynoptic As CommandSynopticCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList =
        objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandSynoptic =
        objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSynoptic.ParameterFile = "Param1.movpar"
objCommandList.SaveChanges

        Set objCommandSynoptic = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
```

```
        Dim objCommandList As CommandsListCmdTarget
         Dim objCommandSynoptic As CommandSynopticCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandSynoptic =
        objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSynoptic.ParameterFile = "Param1.movpar"
        objCommandList.SaveChanges

        Set objCommandSynoptic = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
        objRect = Nothing
End Sub
```

PrintBottomMargin, CommandSynopticCmdTarget Property

Syntax PrintBottomMargin = _Long

Description This property sets or returns the print page's bottom margin size for the referenced Screen Command. This value is in mm and the "-1" setting uses the default print page size.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result Long

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSynoptic As CommandSynopticCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
```

```
        Set objCommandSynoptic = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSynoptic.PrintBottomMargin = 10
        objCommandList.SaveChanges()

        Set objCommandSynoptic = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSynoptic As CommandSynopticCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandSynoptic = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSynoptic.PrintBottomMargin = 10
        objCommandList.SaveChanges()

        Set objCommandSynoptic = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

PrintLeftMargin, CommandSynopticCmdTarget Property

Syntax PrintLeftMargin = _Long

Description This property sets or returns the print page's left margin size for the referenced Screen Command. This value is in mm and the "-1" setting uses the default print page size.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result Long

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSynoptic As CommandSynopticCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandSynoptic = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSynoptic.PrintLeftMargin = 10
        objCommandList.SaveChanges()

        Set objCommandSynoptic = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSynoptic As CommandSynopticCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandSynoptic = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSynoptic.PrintLeftMargin = 10
        objCommandList.SaveChanges()

        Set objCommandSynoptic = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

PrintPageHeight, CommandSynopticCmdTarget Property

Syntax PrintPageHeight = _Long

Description This property sets or returns the print page's height size for the referenced Screen Command. This value is in mms and the "-1" setting uses the default print page size.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the

object's command list.

Parameter	Description
None	None

Result Long

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSynoptic As CommandSynopticCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandSynoptic = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSynoptic.PrintPageHeight = 200
        objCommandList.SaveChanges()

        Set objCommandSynoptic = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSynoptic As CommandSynopticCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandSynoptic = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSynoptic.PrintPageHeight = 200
        objCommandList.SaveChanges()

        Set objCommandSynoptic = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

PrintPageWidth, CommandSynopticCmdTarget Property

Syntax PrintPageWidth = _Long

Description This property sets or returns the print page's width size for the referenced Screen Command. This value is in mms and the "-1" setting uses the default print page size.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result Long

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSynoptic As CommandSynopticCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandSynoptic = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSynoptic.PrintPageWidth = 250
        objCommandList.SaveChanges()

        Set objCommandSynoptic = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSynoptic As CommandSynopticCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandSynoptic = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSynoptic.PrintPageWidth = 250
        objCommandList.SaveChanges()

        Set objCommandSynoptic = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

PrintRightMargin, CommandSynopticCmdTarget Property

Syntax PrintRightMargin = _Long

Description This property sets or returns the print page's right margin size for the referenced Screen Command. This value is in mm and the "-1" setting uses the default print page size.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result Long

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSynoptic As CommandSynopticCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandSynoptic = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSynoptic.PrintRightMargin = 10
        objCommandList.SaveChanges()

        Set objCommandSynoptic = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSynoptic As CommandSynopticCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandSynoptic = objCommandList.GetCommandInterfaceAtPos(0)
```

```
        objCommandSynoptic.PrintRightMargin = 10
        objCommandList.SaveChanges()

        Set objCommandSynoptic = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

PrintTopMargin, CommandSynopticCmdTarget Property

Syntax PrintTopMargin = _Long

Description This property sets or returns the print page's Top margin size for the referenced Screen Command. This value is in mm and the "-1" setting uses the default print page size.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result Long

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSynoptic As CommandSynopticCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandSynoptic = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSynoptic.PrintTopMargin = 10
        objCommandList.SaveChanges()

        Set objCommandSynoptic = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSynoptic As CommandSynopticCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandSynoptic = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSynoptic.PrintTopMargin = 10
        objCommandList.SaveChanges()

        Set objCommandSynoptic = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

ResizeableBorder, CommandSynopticCmdTarget Property

Syntax ResizeableBorder = _Boolean

Description This property, when set to True, allows the Screen window opened by the referenced Screen Command to be resized. This parameter is used only when the "Open Modal" and "Open Frame" commands are used.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result Boolean

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSynoptic As CommandSynopticCmdTarget
```

```
        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList =
        objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandSynoptic =
        objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSynoptic.ResizeableBorder = False
        objCommandList.SaveChanges

        Set objCommandSynoptic = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSynoptic As CommandSynopticCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandSynoptic =
        objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSynoptic.ResizeableBorder = False
        objCommandList.SaveChanges

        Set objCommandSynoptic = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

ShowBorder, CommandSynopticCmdTarget Property

Syntax ShowBorder = _Boolean

Description When set at True this property shows the outside border of the Screen opened with the referenced Screen Command. This parameter is only used when the "Open Modal" and "Open Frame" commands are used.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description

None	None

Result Boolean

Example1:

```
Public Sub Click()
	Dim objButtonRelease As ButtonCmdTarget
	Dim objCommandList As CommandsListCmdTarget
	Dim objCommandSynoptic As CommandSynopticCmdTarget

	Set objButtonRelease =
	GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
	Set objCommandList =
	objButtonRelease.GetCommandsInterfaceOnRelease
	Set objCommandSynoptic =
	objCommandList.GetCommandInterfaceAtPos(0)

	objCommandSynoptic.ShowBorder = False
	objCommandList.SaveChanges

	Set objCommandSynoptic = Nothing
	Set objCommandList = Nothing
	Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
	Dim objRect As DrawCmdTarget
	Dim objCommandList As CommandsListCmdTarget
	Dim objCommandSynoptic As CommandSynopticCmdTarget

	Set objRect = GetSynopticObject.GetSubObject("objRect")
	Set objCommandList = objRect.GetCommandsInterfaceOnRelease
	Set objCommandSynoptic =
	objCommandList.GetCommandInterfaceAtPos(0)

	objCommandSynoptic.ShowBorder= False
	objCommandList.SaveChanges

	Set objCommandSynoptic = Nothing
	Set objCommandList = Nothing
	Set objRect = Nothing
End Sub
```

ShowCaption, CommandSynopticCmdTarget Property

Syntax ShowCaption = _Boolean

Description When set to True, this property allows the Caption to show in the Screen opened by the referenced Screen Command. This parameter is considered only when the "Open Modal" and "Open Frame" command are used.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply

modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result Boolean

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSynoptic As CommandSynopticCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList =
        objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandSynoptic =
        objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSynoptic.ShowCaption = False
        objCommandList.SaveChanges

        Set objCommandSynoptic = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSynoptic As CommandSynopticCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandSynoptic =
        objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSynoptic.ShowCaption = False
        objCommandList.SaveChanges

        Set objCommandSynoptic = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

ShowMaximizedBtn, CommandSynopticCmdTarget Property

Syntax ShowMaximizedBtn = _Boolean

Description When set True this property shows the button for maximizing the Screen opened with the referenced Screen Command. This parameter is considered only when the "Open Modal" and "Open Frame" commands are used. This setting will be ignored if the "System Menu" property had not been enabled.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result Boolean

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSynoptic As CommandSynopticCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList =
        objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandSynoptic =
        objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSynoptic.ShowMaximizedBtn = False
        objCommandList.SaveChanges

        Set objCommandSynoptic = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSynoptic As CommandSynopticCmdTarget
```

```
        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandSynoptic =
        objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSynoptic.ShowMaximizedBtn = False
        objCommandList.SaveChanges



        Set objCommandSynoptic = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

ShowMinimizedBtn, CommandSynopticCmdTarget Property

Syntax ShowMinimizedBtn = _Boolean

Description When set True this property shows the button for minimizing the Screen opened with the referenced Screen Command. This parameter is considered only when the "Open Modal" and "Open Frame" commands are used. This setting will be ignored if the "System Menu" property had not been enabled.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result Boolean

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSynoptic As CommandSynopticCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
```

```
        Set objCommandList =
        objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandSynoptic =
        objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSynoptic.ShowMinimizedBtn = False
        objCommandList.SaveChanges

        Set objCommandSynoptic = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSynoptic As CommandSynopticCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandSynoptic =
        objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSynoptic.ShowMinimizedBtn = False
        objCommandList.SaveChanges

        Set objCommandSynoptic = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

ShowSystemMenu, CommandSynopticCmdTarget Property

Syntax ShowSystemMenu= _Boolean

Description When set True, this property shows the System Menu in the menu bar of the Screen opened with the referenced Screen window. This parameter is only valid when the "Open Modal" or "Open Frame" commands are used. If the Menu bar has not been enabled this setting will be ignored.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description

None	None

Result Boolean

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSynoptic As CommandSynopticCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList =
        objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandSynoptic =
        objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSynoptic.ShowSystemMenu = False
        objCommandList.SaveChanges

        Set objCommandSynoptic = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSynoptic As CommandSynopticCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandSynoptic =
        objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSynoptic.ShowSystemMenu = False
        objCommandList.SaveChanges

        Set objCommandSynoptic = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

Width, CommandSynopticCmdTarget Property

Syntax Width = _Long

Description This property sets or returns the width of the window opened with the referenced Screen Command. This value is expressed in pixels (with the "0" value for default sizes). This parameter is only valid is the "Open Modal" or Open Frame" commands were used.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply

modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result Long

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSynoptic As CommandSynopticCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList =
        objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandSynoptic =
        objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSynoptic.Width = 600
        objCommandList.SaveChanges


        Set objCommandSynoptic = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSynoptic As CommandSynopticCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandSynoptic =
        objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSynoptic.Width = 600
        objCommandList.SaveChanges


        Set objCommandSynoptic = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

XPos, CommandSynopticCmdTarget Property

Syntax XPos = _Long

Description This property sets or returns the left horizontal position of the window opened with the referenced Screen Command. The value is expressed in pixels ("-1" value for default position). This parameter is only valid when the "Open Modal" and "Open Frame" commands have been used.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result Long

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSynoptic As CommandSynopticCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList =
        objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandSynoptic =
        objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSynoptic.XPos = 50
        objCommandList.SaveChanges

        Set objCommandSynoptic = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSynoptic As CommandSynopticCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
```

```
        Set objCommandSynoptic =
        objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSynoptic.XPos = 50
        objCommandList.SaveChanges


        Set objCommandSynoptic = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

YPos, CommandSynopticCmdTarget Property

Syntax YPos = _Long

Description This property sets or returns the top corner vertical position of the window opened with the referenced Screen Command. The value is expressed in pixels ("-1" value for default position). This parameter is only valid when the "Open Modal" and "Open Frame" commands have been used.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result Long

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSynoptic As CommandSynopticCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList =
        objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandSynoptic =
        objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSynoptic.YPos = 50
        objCommandList.SaveChanges
```

```
        Set objCommandSynoptic = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSynoptic As CommandSynopticCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandSynoptic =
        objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSynoptic.YPos = 50
        objCommandList.SaveChanges

        Set objCommandSynoptic = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

20. CommandSystemCmdTarget

GetCommandBaseInterface, CommandSystemCmdTarget Function

Syntax GetCommandBaseInterface()

Description This function gets the CommandBaseCmdTarget interface relating to the referenced command type.

Parameter	Description
None	None

Result Object: returns a CommandBaseCmdTarget object type.

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSystem As CommandSystemCmdTarget
        Dim objCommandBase As CommandBaseCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandSystem = objCommandList.GetCommandInterfaceAtPos(0)

        Set objCommandBase = objCommandSystem.GetCommandBaseInterface

        Set objCommandBase = Nothing
        Set objCommandSystem = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSystem As CommandSystemCmdTarget
        Dim objCommandBase As CommandBaseCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandSystem = objCommandList.GetCommandInterfaceAtPos(0)

        Set objCommandBase = objCommandSystem.GetCommandBaseInterface

        Set objCommandBase = Nothing
        Set objCommandSystem = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
```

```
End Sub
```

Action, CommandSystemCmdTarget Property

Syntax Action= eSystemMode

Description This property sets or returns the action that must execute the referenced System command. The action type can be specified using the eSystemMode enumerator or by inserting the corresponding numeric value:

enum_sm_exitWindow (value 0, Stops Operating System)
enum_sm_exitMovicon (value 1, Stops Application)
enum_sm_RunExec (value 2, Runs Application)
enum_sm_RunExecWait (Value 3, Runs Application and Waits)
enum_sm_PlaySoundFile (value 4, Plays Sound File)
enum_sm_Beep (value 5, executes Beep)
enum_sm_Speak (value 6, Speech)
enum_sm_rebootWindow (value 7, reboots system)
enum_sm_ShowHideTraceBar (value 8, Shows/Hides Output Window)
enum_sm_WaitTime (value 9, time to wait)
enum_ops_capturesave (value 10, Captures and Saves)

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result eSystemMode

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSystem As CommandSystemCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandSystem = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSystem.Action = enum_sm_exitMovicon
        objCommandList.SaveChanges
```

```
        Set objCommandSystem = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSystem As CommandSystemCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandSystem = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSystem.Action = enum_sm_exitMovicon
        objCommandList.SaveChanges


        Set objCommandSystem = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

CommandLine, CommandSystemCmdTarget Property

Syntax CommandLine = _String

Description This property sets or returns the command string to be used for the "Action" type executed by the referenced System Command.

After having adding or modifying a command on the object's command list you will need to execute the CommandsListCmdTarget interface's SaveChanges method to put changes into effect on the object's command list.
Please be reminded that modifications to command lists remain valid only until the object is unloaded from memory (upon screen closure). When the object is next loaded on screen its command list will be restored with the one associated in design mode. However, its is possible to make command list modifications persistent by associating configuration file to the object and then saving it after having made the modifications and saved the command list.

Parameter	Description
None	None

Result String

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSystem As CommandSystemCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandSystem = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSystem.CommandLine = "Calc.exe"
        objCommandList.SaveChanges

        Set objCommandSystem = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSystem As CommandSystemCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandSystem = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSystem.CommandLine = "Calc.exe"
        objCommandList.SaveChanges

        Set objCommandSystem = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

Timeout, CommandSystemCmdTarget Property

Syntax Timeout = _Long

Description This property sets or returns the timeout in milliseconds which Movicon will wait when applications executed with the referenced Screen Command do not respond.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration

file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result Long

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSystem As CommandSystemCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandSystem = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSystem.Timeout = 1000
        objCommandList.SaveChanges


        Set objCommandSystem = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
            Dim objCommandSystem As CommandSystemCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandSystem = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSystem.Timeout = 1000
        objCommandList.SaveChanges


        Set objCommandSystem = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

WorkingPath, CommandSystemCmdTarget Property

Syntax WorkingPath = _String

Description This property sets or returned the working folder to be used for the "Working folder" parameter for the referenced System Command.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result String

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSystem As CommandSystemCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandSystem = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSystem.WorkingPath = "C:\Temp\"
        objCommandList.SaveChanges


        Set objCommandSystem = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandSystem As CommandSystemCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandSystem = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandSystem.WorkingPath = "C:\Temp\"
        objCommandList.SaveChanges


        Set objCommandSystem = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
```

```
End Sub
```

21. CommandUsersCmdTarget

GetCommandBaseInterface, CommandUsersCmdTarget Function

Syntax GetCommandBaseInterface()

Description This function gets the CommandBaseCmdTarget interface relating to the referenced command type.

Parameter	Description
None	None

Result Object: returns a CommandBaseCmdTarget object type.

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandUser As CommandUsersCmdTarget
        Dim objCommandBase As CommandBaseCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandUser = objCommandList.GetCommandInterfaceAtPos(0)

        Set objCommandBase = objCommandUser.GetCommandBaseInterface

        Set objCommandBase = Nothing
        Set objCommandUser = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = NothingEnd Sub
```

Example2:

```
Public Sub Click()
Dim objRect As DrawCmdTarget
Dim objCommandList As CommandsListCmdTarget
        Dim objCommandUser As CommandUsersCmdTarget
        Dim objCommandBase As CommandBaseCmdTarget

Set objRect = GetSynopticObject.GetSubObject("objRect")
Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandUser = objCommandList.GetCommandInterfaceAtPos(0)

        Set objCommandBase = objCommandUser.GetCommandBaseInterface

Set objCommandBase = Nothing
        Set objCommandUser = Nothing
Set objCommandList = Nothing
Set objRect = Nothing
End Sub
```

Action, CommandUsersCmdTarget Property

Syntax Action= eUserMode

Description This property sets or returns the action which executes the referenced Users Command. The action type can be specified using the eUserMode 'enumerator or by inserting the corresponding numberic value:

enum_um_Logon (value 0, Log on)
enum_um_Logoff (value 1, Log off)
enum_um_EditUsers (value 2, Edits Users List)

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result eUserMode

Example:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandUser As CommandUsersCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandUser = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandUser.Action = enum_um_Logon
        objCommandList.SaveChanges


        Set objCommandUser = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandUser As CommandUsersCmdTarget

Set objRect = GetSynopticObject.GetSubObject("objRect")
Set objCommandList = objRect.GetCommandsInterfaceOnRelease
```

```
        Set objCommandUser = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandUser.Action = enum_um_Logon
        objCommandList.SaveChanges


        Set objCommandUser = Nothing
Set objCommandList = Nothing
Set objRect = Nothing
End Sub
```

Level, CommandUsersCmdTarget Property

Syntax Level = _Long

Description This property sets or returns the minimum level which user must have for logging on using the referenced Users Command.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result Long

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandUser As CommandUsersCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandUser = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandUser.Level = 5
        objCommandList.SaveChanges


        Set objCommandUser = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandUser As CommandUsersCmdTarget

Set objRect = GetSynopticObject.GetSubObject("objRect")
Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandUser = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandUser.Level = 5
        objCommandList.SaveChanges


        Set objCommandUser = Nothing
Set objCommandList = Nothing
Set objRect = Nothing
End Sub
```

22. CommandVariableCmdTarget

GetCommandBaseInterface, CommandVariableCmdTarget Function

Syntax GetCommandBaseInterface()

Description This function gets the CommandBaseCmdTarget interface relating to the referenced command type.

Parameter	Description
None	None

Result Object: returns a CommandBaseCmdTarget object type.

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandVariable As CommandVariableCmdTarget
        Dim objCommandBase As CommandBaseCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandVariable = objCommandList.GetCommandInterfaceAtPos(0)

        Set objCommandBase = objCommandVariable.GetCommandBaseInterface

        Set objCommandBase = Nothing
        Set objCommandVariable = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandVariable As CommandVariableCmdTarget
        Dim objCommandBase As CommandBaseCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandVariable = objCommandList.GetCommandInterfaceAtPos(0)

        Set objCommandBase = objCommandVariable.GetCommandBaseInterface

        Set objCommandBase = Nothing
        Set objCommandVariable = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
```

End Sub

Action, CommandVariableCmdTarget Property

Syntax Action= eSetVariableMode

Description This property sets or returns the action which executed the referenced Variable Command. This action type can be specified using the eSetVariableMode enumerator or by inserting the corresponding numeric value:

enum_svm_set (value 0 to Set)
enum_svm_reset (value 1 to Reset)
enum_svm_toggle (value 2 to Toggle)
enum_svm_strobe (value 3 to strobe)
enum_svm_increase (value 4 to Increase)
enum_svm_decrease (value 5 to Decrease)
enum_svm_alphanumeric (value 6 for Alphanumeric Pad)
enum_svm_numeric (value 7 for Numeric Pad)
enum_svm_AppendValue (value 8 to Append Value)
enum_svm_BackValue (value 9 to Remove Value)
enum_svm_SwapPlusMinus (value 10 to change +/- sign)
enum_svm_AppendDecimalMode (value 11 to append ON-OFF Decimal)
enum_svm_MoveValue (valure to transfer Value)
enum_svm_ResetStatistics (value 13 to Reset Statistics)
enum_svm_MoveMinValue (value 14, to transfer Minimum Value)
enum_svm_MoveMaxValue (value 15 to transfer Maximum Value)
enum_svm_MoveAveValue (value 16 to transfer Average Value)
enum_svm_SetStringID (value 17, Sets string ID)

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result eSetVariableMode

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
```

```
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandVariable As CommandVariableCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandVariable = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandVariable.Action = enum_svm_set
        objCommandList.SaveChanges


        Set objCommandVariable = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandVariable As CommandVariableCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandVariable = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandVariable.Action = enum_svm_set
        objCommandList.SaveChanges


        Set objCommandVariable = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

MaxChar, CommandVariableCmdTarget Property

Syntax MaxChar = _Long

Description This property sets or returns the maximum number of chars that can be set in one variable string using the referenced Command Variable.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description

None	None

Result Long

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandVariable As CommandVariableCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandVariable = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandVariable.MaxChar = 10
        objCommandList.SaveChanges


        Set objCommandVariable = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandVariable As CommandVariableCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandVariable = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandVariable.MaxChar = 10
        objCommandList.SaveChanges


        Set objCommandVariable = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

MaxValue, CommandVariableCmdTarget Property

Syntax MaxValue = _String

Description This property sets or returns the maximum value that can be set in one numeric variable using the referenced Command Variable.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to

command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result String

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandVariable As CommandVariableCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandVariable = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandVariable.MaxValue = “100”
        objCommandList.SaveChanges


        Set objCommandVariable = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandVariable As CommandVariableCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandVariable = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandVariable.MaxValue = “100”
        objCommandList.SaveChanges


        Set objCommandVariable = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

MinValue, CommandVariableCmdTarget Property

Syntax MinValue = _String

Description This property sets or returns the minimum value that can be set in one numeric variable using the referenced Command Variable.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result String

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandVariable As CommandVariableCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandVariable = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandVariable.MinValue = “0”
        objCommandList.SaveChanges

        Set objCommandVariable = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandVariable As CommandVariableCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandVariable = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandVariable.MinValue = “0”
        objCommandList.SaveChanges
```

```
        Set objCommandVariable = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

MoveToVariable, CommandVariableCmdTarget Property

Syntax MoveToVariable = _String

Description This property sets or returns the name of the destination variable to which the value is to be moved using the reference Variable Command.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result String

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandVariable As CommandVariableCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandVariable = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandVariable.MoveToVariable = “VAR0002”
        objCommandList.SaveChanges

        Set objCommandVariable = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandVariable As CommandVariableCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandVariable = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandVariable.MoveToVariable = “VAR0002”
        objCommandList.SaveChanges

        Set objCommandVariable = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

PasswordStyle, CommandVariableCmdTarget Property

Syntax PasswordStyle = _Boolean

Description This property sets or returns the referenced Variable command's 'Password Style' property value. The 'Password Style' property allows you to insert the value in the variable using the Alphanumeric Pad in cripted or in transparent mode.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result Boolean

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandVariable As CommandVariableCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButton").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
```

```
        Set objCommandVariable = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandVariable.PasswordStyle = Not objCommandVariable.PasswordStyle
        objCommandList.SaveChanges

        Set objCommandVariable = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandVariable As CommandVariableCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRectangle")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandVariable = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandVariable.PasswordStyle = Not objCommandVariable.PasswordStyle
        objCommandList.SaveChanges

        Set objCommandVariable = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
        End SubNothing
End Sub
```

StrobeTime, CommandVariableCmdTarget Property

Syntax StrobeTime = _Long

Description This property sets or returns the Strobe Time in milliseconds during which the variable will remain set at the value entered in the "Value" property when using the "Strobe" action for referenced Variable Command. When time has expired, the variable will return back to its previously set value.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result Long

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandVariable As CommandVariableCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandVariable = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandVariable.StrobeTime= 1000
        objCommandList.SaveChanges


        Set objCommandVariable = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandVariable As CommandVariableCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandVariable = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandVariable.StrobeTime= 1000
        objCommandList.SaveChanges


        Set objCommandVariable = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

Value, CommandVariableCmdTarget Property

Syntax Value = _String

Description This property sets or returns the value to be inserted in the variable using the referenced Variable Command.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made

persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result String

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandVariable As CommandVariableCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandVariable = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandVariable.Value= “5”
        objCommandList.SaveChanges


        Set objCommandVariable = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandVariable As CommandVariableCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandVariable = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandVariable.Value= “5”
        objCommandList.SaveChanges


        Set objCommandVariable = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

Variable, CommandVariableCmdTarget Property

Syntax Variable = _String

Description This property sets or returns the name of the variable in which the command is to be activated using the referenced Variable Command.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
None	None

Result String

Example1:

```
Public Sub Click()
        Dim objButtonRelease As ButtonCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandVariable As CommandVariableCmdTarget

        Set objButtonRelease =
        GetSynopticObject.GetSubObject("objButtonRelease").GetObjectInterface
        Set objCommandList = objButtonRelease.GetCommandsInterfaceOnRelease
        Set objCommandVariable = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandVariable.Variable= “VAR00001”
        objCommandList.SaveChanges


        Set objCommandVariable = Nothing
        Set objCommandList = Nothing
        Set objButtonRelease = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
        Dim objCommandVariable As CommandVariableCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease
        Set objCommandVariable = objCommandList.GetCommandInterfaceAtPos(0)

        objCommandVariable.Variable= “VAR00001”
        objCommandList.SaveChanges


        Set objCommandVariable = Nothing
        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

23. DBVariableCmdTarget

CreateNewVar, DBVariableCmdTarget Function

Syntax CreateNewVar(_lpszVariableName, _nType, _lpszStructName)

Description This function allows a new variable to be created during project startup phase. Therefore this function can then be used only by the basic script that has been set as the project startup script.
When used in other parts other than the one above,a "Nothing" object value will be returned.
In addition, it would be always best to check that the returned object variable is valid by using the "IsValid" function of "DBVarObjCmdTarget" interface.

Parameter	Description
lpszVariableName As String	Name of variable to be created at runtime. If variable name already exists, an increasing numeric suffix will be added until variable name becomes unique. (*)To create an Array type variable, you will need to specify it in this parameter in this way: “Variable:NumElements” and in the ‘nType’ indicating the array element types.
nType As Integer	New variables can be created in the following supported types: 0 : enum_VAR_TYPE_BIT 1: enum_VAR_TYPE_SIGNBYTE 2: enum_VAR_TYPE_BYTE 3: enum_VAR_TYPE_SIGNWORD 4: enum_VAR_TYPE_WORD 5: enum_VAR_TYPE_SIGNDWORD 6: enum_VAR_TYPE_DWORD 7: enum_VAR_TYPE_FLOAT 8: enum_VAR_TYPE_DOUBLE 9: enum_VAR_TYPE_STRING 10: enum_VAR_TYPE_ARRAY (*) 11: enum_VAR_TYPE_STRUCT
lpszStructName As String	Prototype structure name to be used for creating new structure variables (nType = enum_VAR_TYPE_STRUCT). This parameter is not considered when variable is a different type.

Result Object
A DBVarObjCmdTarget object is returned when function has been executed successfully, otherwise a Nothing object is returned.

Example:

```
Option Explicit
Sub Main
        Dim objVar As DBVarObjCmdTarget
        Set objVar = CreateNewVar("VAR00002", enum_VAR_TYPE_DWORD, "") 'create
        variable
        If Not objVar Is Nothing Then
                If objVar.IsValid Then
```

```
                    objVar.DynamicSettings = "[DRV]Modbus TCPIP.Sta=Default
                    Station|Unit=1|FC=2|SA=0"
                    objVar.SetOPCServerEnabled(True)
            Else
                    Debug.Print Replace("The variable '%s' is not valid!", "%s",
                    objVar.GetName)
            End If
        Else
            Debug.Print "Failed to create the variable 'RuntimeVariable'"
        End If
End Sub
```

GetDynamicVariable, DBVariableCmdTarget Function

Syntax GetDynamicVariable(_lpszDynVariableName)

Description This function allows you to get the name of the dynamic variable created by Movicon by using the name of the OPC Server Tag to which the variable is connected. This variable can be then used once referenced with the 'GetVariableObject' function.
The syntax is the one specified in the "Dynamic" property of a variable:
[OPC]ServerName\GroupName.TagName

Parameter	Description
_lpszDynVariableName	Name of OPC Server tag.

Result String

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget
        Dim sVariableName As String

        sVariableName                                                    =
         GetDynamicVariable("[OPC]Softing.OPCToolboxDemo_ServerDA.1\watch.device
        1.value 1")
        Set objVar = GetVariableObject(sVariableName)
        Do
            objVar.Value = objVar.Value + 1
            If objVar.Value >= 10000 Then objVar.Value = 0
            DoEvents
        Loop Until IsInStoppingMode
End Sub
```

GetNumDynTag, DBVariableCmdTarget Function

Syntax GetNumDynTag

Description Read function of the number of dynamic variables (tags) used in Movicon.

Parameter	Description
None	None

Result Long

Example:

```
Sub Main
        Dim lRet As Long
        lRet = GetNumDynTag()
        MsgBox CStr(lRet), vbOkOnly, "Test GetNumDynTag"
End Sub
```

GetNumStructDefinitions, DBVariableCmdTarget Function

Syntax GetNumStructDefinitions

Description Read function of number of structure prototypes declared in the Movicon Real Time DB.

Parameter	Description
None	None

Result Long

Example:

```
Sub Main
        Dim lRet As Long
        lRet = GetNumStructDefinitions()
        MsgBox CStr(lRet), vbOkOnly, "Test GetNumStructDefinitions"
End Sub
```

GetNumVariables, DBVariableCmdTarget Function

Syntax GetNumVariables

Description Read function of the number of variables declared in the Movicon Real Time DB.

Parameter	Description
None	None

Result Long

Example:

```
Sub Main
        Dim lRet As Long
        lRet = GetNumVariables()
        MsgBox CStr(lRet), vbOkOnly, "Test GetNumVariables"
End Sub
```

GetRealTimeDBADOConn, DBVariableCmdTarget Function

Syntax GetRealTimeDBADOConn()

Description This function gets the ADODB connection relating to the "Real Time ODBC Settings" of the project's variables.

This function is not supported in Windows CE.(If set alwyay returns a 'null' object)

Parameter	Description
None	None

Result Object
If Function has been executed successfully it will retrieve an object of type ADODB.Connection if otherwise Nothing is returned.

Example:

```
Sub Main
        Dim Conn1 As New ADODB.Connection
        Dim Rs1 As New ADODB.Recordset
        Dim contFields As Integer
        Dim sQuery As String
        Set Conn1 = GetRealTimeDBADOConn
        sQuery = "SELECT * FROM RTVar"
        Set Rs1 = CreateObject("ADODB.Recordset")
        Rs1.Open sQuery, Conn1, adOpenForwardOnly, adLockReadOnly, ADODB.adCmdText
        ' Loop per stampare tutti i campi del recorset
        While Not Rs1.EOF
                For contFields = 0 To (Rs1.Fields.Count-1)
                        Debug.Print Rs1.Fields(contFields).Name & " = " &
                        Rs1.Fields(contFields).Value
                Next
                Rs1.MoveNext
        Wend
        Rs1.Close
        Conn1.Close
End Sub
```

GetTraceDBADOConn, DBVariableCmdTarget Function

Syntax GetTraceDBADOConn()

Description This function gets the ADODB connection relating to the "Trace DB Settings" of the project's variables.

If used in Windows CE, this function will always return an ADOCE.connection.3.1. type object. Furthermore, avoid using the "close method" to close ADO connections, otherwise Movicon will no longer be able to access that database.

Parameter	Description
None	None

Result Object
If Function has been executed successfully it will retrieve an object of type ADODB.Connection if otherwise Nothing is returned.

Example:

```
Sub Main
        Dim Conn1 As New ADODB.Connection
        Dim Rs1 As New ADODB.Recordset
        Dim contFields As Integer
        Dim sQuery As String
        Set Conn1 = GetTraceDBADOConn
        sQuery = "SELECT * FROM VAR00001"
        Set Rs1 = CreateObject("ADODB.Recordset")
        Rs1.Open sQuery, Conn1, adOpenForwardOnly, adLockReadOnly, ADODB.adCmdText
        ' Loop per stampare tutti i campi del recorset
        While Not Rs1.EOF
                For contFields = 0 To (Rs1.Fields.Count-1)
                        Debug.Print Rs1.Fields(contFields).Name & " = " &
                        Rs1.Fields(contFields).Value
                Next
                Rs1.MoveNext
        Wend
        Rs1.Close.
        Conn1.Close
End Sub
```

GetTraceDBDSNConnectionString, DBVariableCmdTarget Function

Syntax GetTraceDBDSNConnectionString

Description This function allows you to get the name of the ODBC connection relating to the trace file of the project variables.

Parameter	Description
None	None

Result String

Example:

```
Sub Main
        MsgBox "Collegamento ODBC = " &
        GetTraceDBDSNConnectionString,vbExclamation,GetProjectTitle
End Sub
```

GetVariableAddressInfo, DBVariableCmdTarget Function

Syntax GetVariableAddressInfo(_lpszVariableName, _nType, _nAreaType, _nAddress, _nSubAddress)

Description Gets data Type information, area data, address and sub address of a variable existing in the Movicon Real Time DB. The True or False return value can be used for verifying whether the variable identified by the lpszVariableName parameter exists in the project.
The nType parameter can have the following values:
0 = Bit
1 = Sign Byte (8 Bits with sign)
2 = Byte (8 Bits without sign)
3 = Sign Word (16 Bits with sign)
4 = Word (16 Bits without sign)
5 = Sign DWord (32 Bits with sign)
6 = DWord (32 Bits without sign)
7 = Float (32 bits single precision)
8 = Double (64 Bits double precision)
9 = String (Termination with 0)
10 = Array of Bytes with fixed length
11 = Data Structure

The nArea Type parameter can have the following values:
0 = Input
1 = Flag
2 = Output
3 = Not Shared

Parameter	Description
lpszVariableName As String	Name of variable.
nType	Index indicating data type.
nAreaType	Index indicating dat area.
nAddress	Variable's address.
nSubAddress	Sub address. Number of bits for the bit type variables. Number of bytes when array variables.

Result String

Example:

```
Sub Main
```

```
        Dim lpszVariableName As String
        Dim nType As Integer
        Dim nAreaType As Integer
        Dim nAddress As Long
        Dim nSubAddress As Integer

        lpszVariableName = "VAR00001"
        GetVariableAddressInfo(lpszVariableName, nType, nAreaType, nAddress, nSubAddress)

        MsgBox "Variable Name = " & lpszVariableName & vbLf & _
                "Variable Type = " & nType & vbLf & _
                "Variable Area = " & nAreaType & vbLf & _
                "Variable Address = " & nAddress & vbLf & _
                "Variable SubAddress = " & nSubAddress, vbInformation, GetProjectTitle
End Sub
```

GetVariableDescription, DBVariableCmdTarget Function

Syntax GetVariableDescription(_lpszVariableName)

Description The function gets the description text associated to the variable declared in the project. When creating variables in the Movicon Real Time DB you can assign each one with a description.
The return value is a null string when the variable has not been declared in the project.

Parameter	Description
lpszVariableName As String	Name of variable.

Result String

Example:

```
Sub Main
        Dim sRet As String
        sRet = GetVariableDescription("VAR00001")
        MsgBox sRet, vbOkOnly, "Test GetVariableDescription"
End Sub
```

GetVariableObject, DBVariableCmdTarget Function

Syntax GetVariableObject(_lpszVariableName)

Description Function which gets an object which can be used for managing a variable in the its properties and methods. To be used with the Set command and with a Object variable type or more precisely a DBVarObjCmdTarget type. For further information on the potentialities offers please refer to the DBVarObjCmdTarget Interface help.

Parameter	Description

lpszVariableName As String	Name of variable.

Result Object
If Function has been executed successfully it will retrieve an object of type DBVarObjCmdTarget if otherwise Nothing is returned.

Example:

```
Sub Main
        Dim objRet As DBVarObjCmdTarget
        Set objRet = GetVariableObject("VAR00001")
        MsgBox objRet.GetDescription(), vbOkOnly, "Test GetVariableObject"
        Set objRet = Nothing
End Sub
```

GetVariableSize, DBVariableCmdTarget Function

Syntax GetVariableSize(_lpszVariableName)

Description Read function of the number of bytes occupied by a certain variable declared in the Movicon Real Time DB.

Parameter	Description
lpszVariableName As String	Name of variable.

Result Long

Example:

```
Sub Main
        Dim lRet As Long
        lRet = GetVariableSize("VAR00001")
        MsgBox CStr(lRet), vbOkOnly, "Test GetVariableSize"
End Sub
```

GetVariableValue, DBVariableCmdTarget Function

Syntax GetVariableValue(_lpszVariableName)

Description Read Function of the value contained in a variable from the Movicon Real Time DB, identified by the name in string fromat (lpszVarName parameter).

Parameter	Description
lpszVariableName As String	Name of variable.

Result Variant

Example:

```
Sub Main
        Dim value As Variant
        value = GetVariableValue("VAR00001")
        MsgBox CStr(value), vbOkOnly, "Test GetVariableValue"
End Sub
```

GetXMLSettings, DBVariableCmdTarget Function

Syntax GetXMLSettings

Description This function returns the Movicon Real Time DB settings and Configuration XML text being the contents of the resource file "projectname.movrealtimedb".

Parameter	Description
None	None

Result String

Example:

```
Sub Main
        MsgBox GetXMLSettings,vbInformation,GetProjectTitle
End Sub
```

IsFirstDBInstance, DBVariableCmdTarget Function

Syntax IsFirstDBInstance

Description This function returns the "false" value when the Movicon instance being run has been run following a "Safe" command ("Open in separate Process (Safe Mode)"). Returns "True" in all other cases.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Main
        Dim vResult As Variant
```

```
        vResult = IsFirstDBInstance()
        MsgBox "IsFirstDBInstance = " & vResult,vbInformation,GetProjectTitle
End Sub
```

IsValidVariable, DBVariableCmdTarget Function

Syntax IsValidVariable(_lpszVariableName)

Description This function allows you to verify whether the requested variable is effectively a valid variable belonging to the Movicon Real Time DB.

Parameter	Description
lpszVariableName As String	Name of variable.

Result Boolean

Example:

```
Sub Main
        Dim vResult As Variant

        vResult = IsValidVariable("VAR00001")
        MsgBox "VAR00001 = " & vResult,vbInformation,GetProjectTitle
End Sub
```

PurgeDynTag, DBVariableCmdTarget Function

Syntax PurgeDynTag

Description This property allows you purge dynamic variables in use to free memory space occupied by them. This operation is done automatically when the 'Purge Dynamic Variable Timeout' has expired which can be set up with the PurgeDynTagTimer function described in the chapter on DBVariableCmdTarget

Parameter	Description
None	None

Result Long

Example:

```
Sub Main
        Dim vResult As Variant
         vResult = PurgeDynTag
         MsgBox "PurgeDynTag = " & vResult,vbInformation,GetProjectTitle
End Sub
```

QualityOf, DBVariableCmdTarget Function

Syntax QualityOf(_lpszVariableName)

Description This property consents you to read the of a quality status of a specified variable.
This property is read only.

The returned values are quality values such as from OPC specifications:

0 = OPC_QUALITY_BAD
64 = OPC_QUALITY_UNCERTAIN
192 = OPC_QUALITY_GOOD
4 = OPC_QUALITY_CONFIG_ERROR
8 = OPC_QUALITY_NOT_CONNECTED
12 = OPC_QUALITY_DEVICE_FAILURE
16 = OPC_QUALITY_SENSOR_FAILURE
20 = OPC_QUALITY_LAST_KNOWN
24 = OPC_QUALITY_COMM_FAILURE
28 = OPC_QUALITY_OUT_OF_SERVICE
68 = OPC_QUALITY_LAST_USABLE
80 = OPC_QUALITY_SENSOR_CAL
84 = OPC_QUALITY_EGU_EXCEEDED
88 = OPC_QUALITY_SUB_NORMAL
216 = OPC_QUALITY_LOCAL_OVERRIDE

This property cannot be used in basic script expressions. For instance, a "QualityOf("VAR00001")" expression cannot be assigned to the "Variable Display" property of a display because the basic expressions only support the WinWrap mathematic operators.

Parameter	Description
lpszVariableName as string	Name of the variable whose quality wants to be known.

Result Integer

Example:

```
Public Sub Click()
        MsgBox "Quality of the variable VAR00001 = " & CStr(QualityOf("VAR00001")),
        vbOkOnly, GetProjectTitle
End Sub
```

SetVariableValue, DBVariableCmdTarget Function

Syntax SetVariableValue(_lpszVariableName, _vtValue)

Description This function writes a value (numeric or string) in the Real time DB variable. The variable must be entered by name in the lpszVariable parameter.

Parameter	Description
lpszVariableName As String	Identifier String of the Variable in the Real Time DB.

vtValue As Variant	Value to be written in the variable.

Result Boolean

Example:

```
Sub Main
        Dim bRet As Boolean
        bRet = SetVariableValue("VAR00001", 11)
        MsgBox CStr(bRet), vbOkOnly, "Test SetVariableValue"
End Sub
```

VariableInUse, DBVariableCmdTarget Function

Syntax VariableInUse(_lpszVariableName) = _Boolean

Description This function allows the variable to set "in use" or "not in use". The True value sets the variable to "in use". The False value sets the variable to "not in use".

The "VariableInUse" method cannot be used in relation to "_SysVar_" system variables. This would generate an error when the basic script code, which executed the command, is unloaded from memory. The error message notified is "No error message available".

Parameter	Description
lpszVariableName As String	Name of variable.

Result Boolean

Example:

```
Sub Main
        Dim vResult As Variant

        vResult = VariableInUse("VAR00001")
        MsgBox "VAR00001 InUse = " & vResult,vbInformation,GetProjectTitle
End Sub
```

EnableInUseVarMng, DBVariableCmdTarget Property

Syntax EnableInUseVarMng = _Boolean

Description This property allows you to enable or disable the project's variables in use management.

Warning: if at runtime this property is set to "false", the project will behave as follows:

- The variable value based events in basic scripts will not be executed until the Basic Script is unloaded from the memory and re-loaded
- The screens loaded in memory should be unloaded and re-loaded, otherwise the screen objects will not be updated on the basis of the values of the linked variables

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Main
        MsgBox "EnableInUseVarMng = " & EnableInUseVarMng, vbInformation, GetProjectTitle
End Sub
```

EnableNTSecurityOPCServerTag, DBVariableCmdTarget Property

Syntax EnableNTSecurityOPCServerTag = _Boolean

Description This property allows you to enable or disable the security management in the OPC Server.

This property is not supported in Windows CE.(if used, always returns 'false')

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Main
        MsgBox "EnableNTSecurityOPCServerTag = " & EnableNTSecurityOPCServerTag,
        vbInformation, GetProjectTitle
End Sub
```

OPCServerEnableAEAck, DBVariableCmdTarget Property

Syntax OPCServerEnableAEAck = _Boolean

Description This property allows you to enable or disable the OPC Server AE and therefore the project's alarm and event notification function to all the OPC

clients connected.

This property is not supported in Windows CE. (If set, always returns a 'false')

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Main
        MsgBox "OPCServerEnableAEAck = " & OPCServerEnableAEAck, vbInformation,
        GetProjectTitle
End Sub
```

OPCServerMinImpersonationLevel, DBVariableCmdTarget Property

Syntax OPCServerMinImpersonationLevel = _Long

Description This property allows you to get or set the minimum impersonation level of the OPC Server.

This property is not supported in Windows CE. (If set, always returns a 'zero')

Parameter	Description
None	None

Result Long

Example:

```
Sub Main
        MsgBox "OPCServerMinImpersonationLevel = " &
        OPCServerMinImpersonationLevel,vbInformation,GetProjectTitle
End Sub
```

OPCServerShutdownClientsTimeout, DBVariableCmdTarget Property

Syntax OPCServerShutdownClientsTimeout = _Long

Description This property allows you to get or set the minimum timeout before shutting down the OPC server.

This property is not supported in Windows CE. (If set, always returns a 'zero')

Parameter	Description
None	None

Result Long

Example:

```
Sub Main
        MsgBox "OPCServerShutdownClientsTimeout = " &
        OPCServerShutdownClientsTimeout,vbInformation,GetProjectTitle
End Sub
```

PurgeDynTagTimer, DBVariableCmdTarget Property

Syntax PurgeDynTagTimer = _Long

Description This property allows you to get or set the number of milliseconds after which the unused dynamic variables are removed from the variables in use management.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Main
        MsgBox "Purge Dynamic Tag Timer = " &
        PurgeDynTagTimer,vbInformation,GetProjectTitle
End Sub
```

TraceDBChangerColName, DBVariableCmdTarget Property

Syntax TraceDBChangerColName = _String

Description This property allows you to get or set the name of the "changer Column" in the variable trace table.

This property's modification will be acquired only if done during project development mode, for example using a Symbol's Dropping Code. In this case, the property will be modified in the project statically. Once the project has been started up in Runtime, modification to this property will be ignored.

Parameter	Description
None	None

Result String

Example:

```
Sub Main
        MsgBox "Changer Column Name = " &
        TraceDBChangerColName,vbInformation,GetProjectTitle
End Sub
```

TraceDBDefVarCharPrecision, DBVariableCmdTarget Property

Syntax TraceDBDefVarCharPrecision = _Long

Description This property allows you to get or set the maximum precision for the string columns of the variable trace table. The number set represents the number of string characters.

Parameter	Description
None	None

Result Long

Example:

```
Sub Main
        MsgBox "Char Precision = " &
        TraceDBDefVarCharPrecision,vbInformation,GetProjectTitle
End Sub
```

TraceDBDsn, DBVariableCmdTarget Property

Syntax TraceDBDsn = _String

Description This property is used for reading or setting the ODBC link name that Movicon will use for connecting to the variable Trace Database. Movicon will cerate a DSN for default using the same project name and "_TraceDB" suffix, configured for accessing the specified database using the project's "Default ODBC Plugin". The DSN name will be of this type:

ProjectName_TraceDB

This property can also be used for customizing the **ODBC** link, by creating customized database different to the one defined in the project's "Default ODBC Plugin".

This property can be used in write only project design mode, for example dropping symbol template code. In Runtime, however, even though it can be modified it will not be applied to the TraceDB, which will continue recording using the initial DSN.

Parameter	Description
None	None

Result String

Example:

```
Sub Main
        MsgBox "DSN = " & TraceDBDsn,vbInformation,GetProjectTitle
End Sub
```

TraceDBLocalTimeColName, DBVariableCmdTarget Property

Syntax TraceDBLocalTimeColName = _String

Description This property allows you to get or set the name of the "Local Time Column" of the variable Trace table.

This property's modification will be acquired only if done during project development mode, for example using a Symbol's Dropping Code. In this case, the property will be modified in the project statically. Once the project has been started up in Runtime, modification to this property will be ignored.

Parameter	Description
None	None

Result String

Example:

```
Sub Main
        MsgBox "Time Column Name = " &
        TraceDBLocalTimeColName,vbInformation,GetProjectTitle
End Sub
```

TraceDBMaxCacheBeforeFlush, DBVariableCmdTarget Property

Syntax TraceDBMaxCacheBeforeFlush = _Long

Description This property allows you to get or set the maximum Cache size before the system unloads the data on the variable Trace file. The number set is expressed in Bytes.

Parameter	Description
None	None

Result Long

Example:

```
Sub Main
        MsgBox "Time Max Cache = " &
        cstr(TraceDBMaxCacheBeforeFlush),vbInformation,GetProjectTitle
End Sub
```

TraceDBMaxError, DBVariableCmdTarget Property

Syntax TraceDBMaxError = _Long

Description This property allows you to get or set the maximum number of DBMS errors after which the connection to the variable Trace database is considered invalid and the data is saved on file in ASCII format in the folder relating to the project ("LOGS").

Parameter	Description
None	None

Result Long

Example:

```
Sub Main
        MsgBox "Max Error = " & TraceDBMaxError,vbInformation,GetProjectTitle
End Sub
```

TraceDBMaxNumberTrans, DBVariableCmdTarget Property

Syntax TraceDBMaxNumberTrans = _Long

Description This property allows you to get or set the maximum number of transitions for each cycle to be updated before being disconnected from the variable Trace database.

Parameter	Description
None	None

Result Long

Example:

```
Sub Main
        MsgBox "Max Num Transition = " &
        TraceDBMaxNumberTrans,vbInformation,GetProjectTitle
End Sub
```

TraceDBMSecColName, DBVariableCmdTarget Property

Syntax TraceDBMSecColName = _String

Description This property allows you to get or set the name of the "MSec Column" of the variable Trace table.

This property's modification will be acquired only if done during project development mode, for example using a Symbol's Dropping Code. In this case, the property will be modified in the project statically. Once the project has been started up in Runtime, modification to this property will be ignored.

Parameter	Description
None	None

Result String

Example:

```
Sub Main
        MsgBox "MSec Column Name = " & TraceDBMSecColName,vbInformation,GetProjectTitle
End Sub
```

TraceDBQualityColName, DBVariableCmdTarget Property

Syntax TraceDBQualityColName = _String

Description This property allows you to get or set the name of the "Quality Column" of the variable Trace table.

This property's modification will be acquired only if done during project development mode, for example using a Symbol's Dropping Code. In this case, the property will be modified in the project statically. Once the project has been started up in Runtime, modification to this property will be ignored.

Parameter	Description
None	None

Result String

Example:

```
Sub Main
        MsgBox "Quality Column Name = " &
        TraceDBQualityColName,vbInformation,GetProjectTitle
End Sub
```

TraceDBRecycleDBConnection, DBVariableCmdTarget Property

Syntax TraceDBRecycleDBConnection = _Boolean

Description This property allows to enable/disable the "Keep the DB Connection open" of the variable Trace database.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Main
        MsgBox "Keep the DB Connection open = " &
        TraceDBRecycleDBConnection,vbInformation,GetProjectTitle
End Sub
```

TraceDBTimeColName, DBVariableCmdTarget Property

Syntax TraceDBTimeColName = _String

Description This propery allows you to get or set the name of the "Tiem Column" of the variable Trace database.

This property's modification will be acquired only if done during project development mode, for example using a Symbol's Dropping Code. In this case, the property will be modified in the project statically. Once the project has been started up in Runtime, modification to this property will be ignored.

Parameter	Description
None	None

Result String

Example:

```
Sub Main
        MsgBox "Time Column Name = " & TraceDBTimeColName,vbInformation,GetProjectTitle
End Sub
```

TraceDBTimeStampColName, DBVariableCmdTarget Property

Syntax TraceDBTimeStampColName = _String

Description This property allows you to read or set the name of the Variable TraceDB table's "TimeStamp Column".

This property's modification will be acquired only if done during project development mode, for example using a Symbol's Dropping Code. In this case, the property will be modified in the project statically. Once the project has been started up in Runtime, modification to this property will be ignored.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim TraceDBTable As DBVariableCmdTarget

        Set TraceDBTable = GetRealTimeDB()
        MsgBox "TraceDBTimeStampColName = " & TraceDBTable.TraceDBTimeStampColName,vbInformation,GetProjectTitle
        Set TraceDBTable = Nothing
End Sub
```

TraceDBUser, DBVariableCmdTarget Property

Syntax TraceDBUser = _String

Description This property allows you to enter the name of the user that will be used for the ODBC connection for the variable Trace database file.

Parameter	Description
None	None

Result String

Example:

```
Sub Main
        MsgBox "User Name = " & TraceDBUser,vbInformation,GetProjectTitle
End Sub
```

TraceDBUserColName, DBVariableCmdTarget Property

Syntax TraceDBUserColName = _String

Description This property allows you to get or set the name of the "User Column" of the variable Trace table.

This property's modification will be acquired only if done during project development mode, for example using a Symbol's Dropping Code. In this case, the property will be modified in the project statically. Once the project has been started up in Runtime, modification to this property will be ignored.

Parameter	Description
None	None

Result String

Example:

```
Sub Main
        MsgBox "User Column Name = " & TraceDBUserColName,vbInformation,GetProjectTitle
End Sub
```

TraceDBValueAfterColName, DBVariableCmdTarget Property

Syntax TraceDBValueAfterColName = _String

Description This property allows you to get or set the name of the "Value After Column" of the variable Trace table.

This property's modification will be acquired only if done during project development mode, for example using a Symbol's Dropping Code. In this case, the property will be modified in the project statically. Once the project has been started up in Runtime, modification to this property will be ignored.

Parameter	Description
None	None

Result String

Example:

```
Sub Main
        MsgBox "Value After Column Name = " &
        TraceDBValueAfterColName,vbInformation,GetProjectTitle
End Sub
```

TraceDBValueBeforeColName, DBVariableCmdTarget Property

Syntax TraceDBValueBeforeColName = _String

Description This property allows you to get or set the name of the "Value before Column " of the variable Trace table.

This property's modification will be acquired only if done during project development mode, for example using a Symbol's Dropping Code. In this case, the property will be modified in the project statically. Once the project has been started up in Runtime, modification to this property will be ignored.

Parameter	Description
None	None

Result String

Example:

```
Sub Main
        MsgBox "Value Before Column Name = " &
        TraceDBValueBeforeColName,vbInformation,GetProjectTitle
```

```
End Sub
```

TraceDBValueColName, DBVariableCmdTarget Property

Syntax TraceDBValueColName = _String

Description This property allows you to read or set the name of the "Value Column" of the variable Trace table.

This property's modification will be acquired only if done during project development mode, for example using a Symbol's Dropping Code. In this case, the property will be modified in the project statically. Once the project has been started up in Runtime, modification to this property will be ignored.

Parameter	Description
None	None

Result String

Example:

```
Sub Main
        MsgBox "Value Column Name = " & TraceDBValueColName,vbInformation,GetProjectTitle
End Sub
```

TraceDBVarDescNameColName, DBVariableCmdTarget Property

Syntax TraceDBVarDescNameColName = _String

Description This property allows you to read or set the name of the variable Trace table's "Variable Description Column".

This property's modification will be acquired only if done during project development mode, for example using a Symbol's Dropping Code. In this case, the property will be modified in the project statically. Once the project has been started up in Runtime, modification to this property will be ignored.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim TraceDBTable As DBVariableCmdTarget

        Set TraceDBTable = GetRealTimeDB()
        MsgBox "TraceDBVarDescNameColName = " &
        TraceDBTable.TraceDBVarDescNameColName,vbInformation,GetProjectTitle
        Set TraceDBTable = Nothing
End Sub
```

TraceDBVarGroupNameColName, DBVariableCmdTarget Property

Syntax TraceDBVarGroupNameColName = _String

Description This property allows you to read or set the name of the variable Trace table's "Variable Group Column".

This property's modification will be acquired only if done during project development mode, for example using a Symbol's Dropping Code. In this case, the property will be modified in the project statically. Once the project has been started up in Runtime, modification to this property will be ignored.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim TraceDBTable As DBVariableCmdTarget

        Set TraceDBTable = GetRealTimeDB()
        MsgBox "TraceDBVarGroupNameColName = " &
        TraceDBTable.TraceDBVarGroupNameColName,vbInformation,GetProjectTitle
        Set TraceDBTable = Nothing
End Sub
```

TraceDBVarNameColName, DBVariableCmdTarget Property

Syntax TraceDBVarNameColName = _String

Description This property allows you to read or set the name of the variable Trace table's "Variable Name Column".

This property's modification will be acquired only if done during project development mode, for example using a Symbol's Dropping Code. In this case, the property will be modified in the project statically. Once the project

has been started up in Runtime, modification to this property will be ignored.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim TraceDBTable As DBVariableCmdTarget

        Set TraceDBTable = GetRealTimeDB()
        MsgBox "TraceDBVarNameColName = " &
        TraceDBTable.TraceDBVarNameColName,vbInformation,GetProjectTitle
        Set TraceDBTable = Nothing
End Sub
```

TraceUseIMDB,DBVariableCmdTarget Property

Syntax TraceUseIMDB = _Boolean

Description This property allows you to read the value from the 'Use IMDB manager" property for recording Traced Variable data.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Main
        MsgBox "Use IMDB manager for Trace DB =  " & TraceUseIMDB, vbInformation,
        GetProjectTitle
End Sub
```

UseSharedDynTag, DBVariableCmdTarget Property

Syntax UseSharedDynTag = _Boolean

Description This function allows you to verify whether the requested variable is effectively a valid variable belonging to the Movicon Real Time DB.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Main
        MsgBox "UseSharedDynTag = " & UseSharedDynTag,vbInformation,GetProjectTitle
End Sub
```

24. DBVarObjCmdTarget

GetAccessLevelReadMask, DBVarObjCmdTarget Function

Syntax GetAccessLevelReadMask

Description This function returns a number that, interpreted in bit, describes the access levels to the variable for granting access in read. The value is Long type but the information is contained in the most significant word. The actual number of access levels managed is 16 corresponding to the most significant 16 bit.

Parameter	Description
None	None

Result Long

Example:

```
Sub Main
        Dim objRet As DBVarObjCmdTarget
        Dim lReadMask As Long
        Dim lWriteMask As Long
        Set objRet = GetVariableObject("VAR00001")
        lReadMask = objRet.GetAccessLevelReadMask()
        lWriteMask = objRet.GetAccessLevelWriteMask()
        MsgBox "Read Mask = " & Left(Hex(lReadMask),4) & vbCrLf & _
        "Write Mask = " & Left(Hex(lWriteMask),4), _
        vbOkOnly, "Test Mask"
        Set objRet = Nothing
End Sub
```

GetAccessLevelWriteMask, DBVarObjCmdTarget Function

Syntax GetAccessLevelWriteMask

Description This function returns a number that, interpreted in bit, describes the access levels to the variable for granting access in write. The value is Long type but the information is contained in the most significant word. The actual number of access levels managed is 16 corresponding to the most significant 16 bit.

Parameter	Description
None	None

Result Long

Example:

```
Sub Main
        Dim objRet As DBVarObjCmdTarget
        Dim lReadMask As Long
        Dim lWriteMask As Long
        Set objRet = GetVariableObject("VAR00001")
        lReadMask = objRet.GetAccessLevelReadMask()
        lWriteMask = objRet.GetAccessLevelWriteMask()
        MsgBox "Read Mask = " & Left(Hex(lReadMask),4) & vbCrLf & _
        "Write Mask = " & Left(Hex(lWriteMask),4), _
        vbOkOnly, "Test Mask"
        Set objRet = Nothing
End Sub
```

GetAddress, DBVarObjCmdTarget Function

Syntax GetAddress

Description This function returns a number which identified the absolute address in byte of the variable declared in the project's Real Time DB in reference to the associated data area (Input, Output or Flag).
When dealing with a non shared variable (without any assigned absolute address) the value will return '0'. When dealing with bit type variables, the returned address will not specify the bit's position within the byte; in order to get the exact bit address you need to use the GetBitNumber() function.

Parameter	Description
None	None

Result Long

Example:

```
Sub Main
        Dim objRet As DBVarObjCmdTarget
        Set objRet = GetVariableObject("VAR00001")
        MsgBox CStr(objRet.GetAddress()), vbOkOnly,GetProjectTitle
        Set objRet = Nothing
End Sub
```

GetAlarmListName, DBVarObjCmdTarget Function

Syntax GetAlarmListName(_nIndex)

Description This function returns the name of the alarm numbered in the parameter relating to the reference variable.

Parameter	Description
nIndex as Long	indexed alarm number on the list relating to the reference variable

Result String

Example:

```
Sub Main
        Dim objRet As DBVarObjCmdTarget
        Set objRet = GetVariableObject("VAR00001")
        If objRet.GetAlarmListNum()>0 Then
                MsgBox "GetAlarmListName: " & CStr(objRet.GetAlarmListName(0)), vbOkOnly,
                GetProjectTitle
        End If
        Set objRet = Nothing
End Sub
```

GetAlarmListNum, DBVarObjCmdTarget Function

Syntax GetAlarmListNum

Description The function returns the number of the alarms relating to the reference variable.

Parameter	Description
None	None

Result Long

Example:

```
Sub Main
        Dim objRet As DBVarObjCmdTarget
        Set objRet = GetVariableObject("VAR00001")
        MsgBox "GetAlarmListNum: " & CStr(objRet.GetAlarmListNum()), vbOkOnly,
        GetProjectTitle
        Set objRet = Nothing
End Sub
```

GetAlarmObject, DBVarObjCmdTarget Function

Syntax GetAlarmObject(_lpszAlarmName)

Description This function returns the alarm object identified by its name in string format (lpszAlarmName parameter). In cases where the alarm has been associated to the variable you will have to specify not only the alarm's name but also the name of the variable:

GetAlarmObject(<Alarm Name> <Variable Name>)

Parameter	Description
lpszAlarmName as String	name of the alarm to be retrieved.

Result Object
If Function has been executed successfully it will retrieve an object of type AlarmCmdTarget if otherwise Nothing is returned.

Example1:

```
Public Sub Click()
        Dim objRet As DBVarObjCmdTarget
        Dim objAlarm As AlarmCmdTarget

        Set objRet = GetVariableObject("VAR00001")
        Set objAlarm = objRet.GetAlarmObject("Alarm00001") 'where Alarmr00001 is
        the alarm set by VAR00001
        If Not objAlarm Is Nothing Then
                Debug.Print objAlarm.Name
                Set objAlarm = Nothing
        End If
        Set objRet = Nothing
End Sub
```

Example2:

```
Public Sub Click()
        Dim objRet As DBVarObjCmdTarget
        Dim objAlarm As AlarmCmdTarget

        Set objRet = GetVariableObject("VAR00001")
        Set objAlarm = objRet.GetAlarmObject("Alarm00001 VAR00001") 'where
        Alarmr00001 is the alarm associate to VAR00001
        If Not objAlarm Is Nothing Then
                Debug.Print objAlarm.Name
                Set objAlarm = Nothing
        End If
        Set objRet = Nothing
End Sub
```

GetAreaType, DBVarObjCmdTarget Function

Syntax GetAreaType

Description This function returns a number that identifies the data area type of the variable declared in the project.

The returned value are:

0 = enum_VAR_AREA_INPUT
1 = enum_VAR_AREA_FLAG
2 = enum_VAR_AREA_OUTPUT
-2147220992 = enum_VAR_AREA_TYPE_E_UNKNOWN

Parameter	Description
None	None

Result enum eVariableAreaType

Example:

```
Sub Main
        Dim objRet As DBVarObjCmdTarget
        Set objRet = GetVariableObject("VAR00001")
        MsgBox CStr(objRet.GetAreaType()), vbOkOnly, GetProjectTitle
        Set objRet = Nothing
End Sub
```

GetBitNumber, DBVarObjCmdTarget Function

Syntax GetBitNumber

Description This function returns a number, for a bit type variable with an absolute address (Flag, Input, Output area), that identifies its position in the byte to the same absolute address. The position may obtain values from 0 to 7. The variable's absolute address can be retrieved by using the GetAddress() function.
When dealing with a non shared variable (without an assigned absolute address) or a type different from the Bit, the returned value will be 0.

Parameter	Description
None	None

Result Integer

Example:

```
'FirstBit has absolute address 3.2
'SecondBit has absolute address 3.5
Public Sub Click()
        Dim objRet1 As DBVarObjCmdTarget
        Dim objRet2 As DBVarObjCmdTarget
        Set objRet1 = GetVariableObject("FirstBit")
        Set objRet2 = GetVariableObject("SecondBit")
        'GetAddress return 3 - GetBitNumber return 2
        MsgBox "FirstBit has address " & CStr(objRet1.GetAddress()) & " and the position is " &
        CStr(objRet1.GetBitNumber()), vbOkOnly, GetProjectTitle
        'GetAddress return 3 - GetBitNumber return 5
        MsgBox "SecondBit has address " & CStr(objRet2.GetAddress()) & " and the position is " &
        CStr(objRet2.GetBitNumber()), vbOkOnly, GetProjectTitle
        Set objRet1 = Nothing
        Set objRet2 = Nothing
End Sub
```

GetDataLoggerListNum, DBVarObjCmdTarget Function

Syntax GetDataLoggerListNum

Description This function returns the number of data loggers relevant to the reference variable.

Parameter	Description
None	None

Result Long

Example:

```
Sub Main
        Dim objRet As DBVarObjCmdTarget
        Set objRet = GetVariableObject("VAR00001")
        MsgBox "GetDataLoggerListNum: " & CStr(objRet.GetDataLoggerListNum()), vbOkOnly,
        GetProjectTitle
        Set objRet = Nothing
End Sub
```

GetDataLoggerName, DBVarObjCmdTarget Function

Syntax GetDataLoggerName(_nIndex)

Description This function returns the name of the data logger numbered on the list in which the variable is recorded.

Parameter	Description
nIndex as Long	Data logger list number in which the value of the reference variable is recorded.

Result String

Example:

```
Sub Main
        Dim objRet As DBVarObjCmdTarget
        Dim numObj As Variant
        Set objRet = GetVariableObject("VAR00001")
        numObj = objRet.GetDataLoggerListNum()
        If numObj<>0 Then
        MsgBox "DataLoggerName 0: " & objRet.GetDataLoggerName(0),
        vbOkOnly,GetProjectTitle
        End If
        Set objRet = Nothing
End Sub
```

Description, DBVarObjCmdTarget Function

Syntax Description = _String

Description This function sets or returns the description text associated to the referenced variable. When creating variables in the Movicon RealTime DB you can also assign them a description text.

Parameter	Description
None	None

Result String

Example:

```
Sub Main
        Dim objRet As DBVarObjCmdTarget
        Set objRet = GetVariableObject("VAR00001")
        MsgBox objRet.Description(), vbOkOnly, GetProjectTitle
        Set objRet = Nothing
End Sub
```

GetEventListName, DBVarObjCmdTarget Function

Syntax GetEventListName(_nIndex)

Description This function returns the event's list name relating to the reference variable.

Parameter	Description
nIndex as Long	event's list number relating to the reference variable.

Result String

Example:

```
Sub Main
        Dim objRet As DBVarObjCmdTarget
        Set objRet = GetVariableObject("VAR00001")
        If objRet.GetEventListNum()>0 Then
                MsgBox "GetEventListName: " & CStr(objRet.GetEventListName(0)), vbOkOnly,
                GetProjectTitle
        End If
        Set objRet = Nothing
End Sub
```

GetEventListNum, DBVarObjCmdTarget Function

Syntax GetEventListNum

Description This function returns the event's list number relating to the reference variable.

Parameter	Description
None	None

Result Long

Example:

```
Sub Main
        Dim objRet As DBVarObjCmdTarget
        Set objRet = GetVariableObject("VAR00001")
        MsgBox "GetAlarmListNum: " & CStr(objRet.GetEventListNum()), vbOkOnly,
        GetProjectTitle
        Set objRet = Nothing
End Sub
```

GetEventObject, DBVarObjCmdTarget Function

Syntax GetEventObject

Description This function returns the event object identified by its name in string format (lpszEventName parameter).

Parameter	Description
lpszEventName as String	name of event to be retrieved.

Result Object
If Function has been executed successfully it will retrieve an object of type EventCmdTarget if otherwise Nothing is returned.

Example:

```
Public Sub Click()
        Dim objRet As DBVarObjCmdTarget
        Dim objEvent As EventCmdTarget
        Set objRet = GetVariableObject("VAR00001")
        Set objEvent = objRet.GetEventObject("EventoVar00001") 'supposed event name
        associated to VAR00001 is EventoVar00001
        If Not objEvent Is Nothing Then
        Debug.Print objEvent .Enabled
        Set objEvent = Nothing
        End If
        Set objRet = Nothing
End Sub
```

GetInitialTimeInUse, DBVarObjCmdTarget Function

Syntax GetInitialTimeInUse

Description This function returns the data and time in which the variable went in use in the project.

Parameter	Description
None	None

Result Date

Example:

```
Sub Main
        Dim objRet As DBVarObjCmdTarget
        Dim dateInitUse As Date
        Dim dateLastUse As Date
        Dim dateNotInUse As Date
        Set objRet = GetVariableObject("VAR00001")
        dateInitUse = objRet.GetInitialTimeInUse()
        dateLastUse = objRet.GetLastTimeInUse()
        dateNotInUse = objRet.GetTimeNotInUse()
        MsgBox "Init = " & Format(dateInitUse,"hh.nn.ss - dd/mm/yyyy") & vbCrLf & _
                "Last = " & Format(dateLastUse,"hh.nn.ss - dd/mm/yyyy") & vbCrLf & _
                "Not in use = " & Format(dateNotInUse,"hh.nn.ss - dd/mm/yyyy"), _
                vbOkOnly, "Test Use"
        Set objRet = Nothing
End Sub
```

GetInUseCount, DBVarObjCmdTarget Function

Syntax GetInUseCount

Description This function returns the number of objects which use the reference variable.

Parameter	Description
None	None

Result Long

Example:

```
Sub Main
        Dim objRet As DBVarObjCmdTarget
        Set objRet = GetVariableObject("VAR00001")
        MsgBox CStr(objRet.GetInUseCount()), vbOkOnly,GetProjectTitle
        Set objRet = Nothing
End Sub
```

GetInUseObjectAt, DBVarObjCmdTarget Function

Syntax GetInUseObjectAt(_nIndex)

Description This function returns the object which uses the reference variable by the number specified in the parameter.

Parameter	Description
nIndex as Long	list number of object with reference variable in use.

Result Object
If Function has been executed successfully it will retrieve an object of type DrawCmdTarget if otherwise Nothing is returned.

Example:

```
Public Sub Click()
        Dim objRet As DBVarObjCmdTarget
        Dim nobj As Variant
        Dim objRet1 As DrawCmdTarget
        Set objRet = GetVariableObject("VAR1")
        If Not objRet Is Nothing Then
                Debug.Print "InUseCount is -> " & objRet.GetInUseCount()
                nobj = objRet.GetInUseCount()
                For i = 1 To nobj Step 1
                On Error Resume Next
                Set objRet1 = objRet.GetInUseObjectAt(i)
                If Not objRet1 Is Nothing Then
                        'do somthing with object
                        Debug.Print "Object_i name is -> " & objRet1.ObjectName
                Set objRet1 = Nothing
        End If
        On Error GoTo 0
        Next i
        Set objRet = Nothing
        End If
End Sub
```

GetInUseObjectNameAt, DBVarObjCmdTarget Function

Syntax GetInUseObjectNameAt(_nIndex)

Description This function returns the name of the desired listed object which has the reference variable in use.

Parameter	Description
nIndex	list number of object with the reference variable in use.

Result Long

Example:

```
Sub Main
        Dim objRet As DBVarObjCmdTarget
        Dim nobj As Variant
        Set objRet = GetVariableObject("VAR00001")
        nobj = objRet.GetInUseCount()
        MsgBox "In uso " & CStr(objRet.GetInUseCount())&" volte", vbOkOnly, GetProjectTitle
        For i = 0 To (nobj-1) Step 1
        MsgBox CStr(objRet.GetInUseObjectNameAt(i)), vbOkOnly, GetProjectTitle
        Next i
        Set objRet = Nothing
End Sub
```

GetLastTimeInUse, DBVarObjCmdTarget Function

Syntax GetLastTimeInUse

Description This function returns the date and time in which the variable was used for the last time in the project.

Parameter	Description
None	None

Result Date

Example:

```
Sub Main
        Dim objRet As DBVarObjCmdTarget
        Dim dateInitUse As Date
        Dim dateLastUse As Date
        Dim dateNotInUse As Date
        Set objRet = GetVariableObject("VAR00001")
        dateInitUse = objRet.GetInitialTimeInUse()
        dateLastUse = objRet.GetLastTimeInUse()
        dateNotInUse = objRet.GetTimeNotInUse()
        MsgBox "Init = " & Format(dateInitUse,"hh.nn.ss - dd/mm/yyyy") & vbCrLf & _
                "Last = " & Format(dateLastUse,"hh.nn.ss - dd/mm/yyyy") & vbCrLf & _
                "Not in use = " & Format(dateNotInUse,"hh.nn.ss - dd/mm/yyyy"), _
                vbOkOnly, "Test Use"
        Set objRet = Nothing
End Sub
```

GetMemberObjectFromIndex, DBVarObjCmdTarget Function

Syntax GetMemberObjectFromIndex(_nIndex)

Description This function allows you to get the Member variable within a variable structure type starting from the position passed as parameter within. When the start variable is not structure type or the position passed as parameter is not a defined member variable, the Nothing object is returned.

Parameter	Description
nIndex As Long	Member name.

Result Object
If Function has been executed successfully it will retrieve an object of type DBVarObjCmdTarget if otherwise Nothing is returned.

Example:

```
Public Sub Click()
        Dim objRet As DBVarObjCmdTarget
        Dim Member0 As DBVarObjCmdTarget
        Set objRet = GetVariableObject("_SysVar_")
        Set Member0 = objRet.GetMemberObjectFromIndex(0)
        If Not Member0 Is Nothing Then
                Debug.Print "Member0 name is -> " & Member0.GetName 'return SimSinDouble
        Set Member0 = Nothing
        End If
        Set objRet = Nothing
End Sub
```

GetMemberObjectFromName, DBVarObjCmdTarget Function

Syntax GetMemberObjectFromName(_lpszMember)

Description This function allows you to get the member variable within a variable structure type starting with the name of the member variable passed as parameter. When the start variable is not a structure type or the name passed as parameter is not referred to any member variable, the Nothing object is returned.

Parameter	Description
lpszMember As String	Member Name.

Result Object
If Function has been executed successfully it will retrieve an object of type DBVarObjCmdTarget if otherwise Nothing is returned.

Example:

```
Public Sub Click()
        Dim objRet As DBVarObjCmdTarget
        Dim Member0 As DBVarObjCmdTarget
        Set objRet = GetVariableObject("_SysVar_")
        Set Member0 = objRet.GetMemberObjectFromName("SimSinDouble")
        If Not Member0 Is Nothing Then
        Debug.Print "Member0 type is -> " & Member0.GetType 'return enum_VAR_TYPE_DOUBLE
        = 8
        Set Member0 = Nothing
        End If
        Set objRet = Nothing
End Sub
```

GetName, DBVarObjCmdTarget Function

Syntax GetName

Description This function returns the symbolic name of the variable declared in the project.

Parameter	Description
None	None

Result String

Example:

```
Sub Main
        Dim objRet As DBVarObjCmdTarget
        Set objRet = GetVariableObject("VAR00001")
        MsgBox objRet.GetName(), vbOkOnly,GetProjectTitle
        Set objRet = Nothing
End Sub
```

GetNumObjectsInHeap, DBVarObjCmdTarget Function

Syntax GetNumObjectsInHeap

Description This function returns the number of objects in the memory heap with variable reference.

Parameter	Description
None	None

Result Long

Example:

```
Sub Main
        Dim objRet As DBVarObjCmdTarget
        Set objRet = GetVariableObject("VAR00001")
        MsgBox CStr(objRet.GetNumObjectsInHeap()), vbOkOnly, "Test GetNumObjectsInHeap"
        Set objRet = Nothing
End Sub
```

GetStructName, DBVarObjCmdTarget Function

Syntax GetStructName

Description This function returns the structure prototype's symbolic name of the variable declared in the project. When the reference variable is not a structure type, the string will be returned empty.

Parameter	Description
None	None

Result String

Example:

```
Sub Main
        Dim objRet As DBVarObjCmdTarget
        Set objRet = GetVariableObject("VAR00001")
        If CStr(objRet.GetType()) = 11 Then ' 11 = Struct type
                MsgBox CStr(objRet.GetStructName()), vbOkOnly, "Test GetStructName"
        End If
        Set objRet = Nothing
End Sub
```

GetStructParentObject, DBVarObjCmdTarget Function

Syntax GetStructParentObject

Description This function returns the pointer to the structure object relating to the reference variable.

Parameter	Description
None	None

Result Object
If Function has been executed successfully it will retrieve an object of type DBVarObjCmdTarget if otherwise Nothing is returned.

Example:

```
Sub Main
        Dim objRet As DBVarObjCmdTarget
        Dim objParent as DBVarObjCmdTarget
        Set objRet = GetVariableObject("VAR00001")
        If CStr(objRet.GetType()) = 11 Then ' 11 = Struct type
                Set objParent = objRet.GetStructParentObject()
                'do something
        End If
        Set objParent = Nothing
        Set objRet = Nothing
End Sub
```

GetTimeNotInUse, DBVarObjCmdTarget Function

Syntax GetTimeNotInUse

Description This returns the date and time in which the variable ceased to be used in the project (the last time the project was stopped).

Parameter	Description
None	None

Result Date

Example:

```
Sub Main
        Dim objRet As DBVarObjCmdTarget
        Dim dateInitUse As Date
        Dim dateLastUse As Date
        Dim dateNotInUse As Date
        Set objRet = GetVariableObject("VAR00001")
        dateInitUse = objRet.GetInitialTimeInUse()
        dateLastUse = objRet.GetLastTimeInUse()
        dateNotInUse = objRet.GetTimeNotInUse()
        MsgBox "Init = " & Format(dateInitUse,"hh.nn.ss - dd/mm/yyyy") & vbCrLf & _
                "Last = " & Format(dateLastUse,"hh.nn.ss - dd/mm/yyyy") & vbCrLf & _
                "Not in use = " & Format(dateNotInUse,"hh.nn.ss - dd/mm/yyyy"), _
                vbOkOnly, "Test Use"
        Set objRet = Nothing
End Sub
```

GetTimeStamp, DBVarObjCmdTarget Function

Syntax GetTimeStamp

Description This function returns the date and time of the last variable update.

Parameter	Description
None	None

Result Date

Example:

```
Sub Main
        Dim objRet As DBVarObjCmdTarget
```

```
        Dim dTimeStamp As Date

        Set objRet = GetVariableObject("VAR00001")
        dTimeStamp = objRet.GetTimeStamp()
        MsgBox "TimeStamp = " & Format(dTimeStamp,"hh.nn.ss - dd/mm/yyyy"),vbOkOnly, "Test Use"
        Set objRet = Nothing
End Sub
```

GetType, DBVarObjCmdTarget Function

Syntax GetType()

Description This function returns the number which identifies the type of variable declared in the project.

The returned value are:

0 = enum_VAR_TYPE_BIT
1 = enum_VAR_TYPE_SIGNBYTE
2 = enum_VAR_TYPE_BYTE
3 = enum_VAR_TYPE_SIGNWORD
4 = enum_VAR_TYPE_WORD
5 = enum_VAR_TYPE_SIGNDWORD
6 = enum_VAR_TYPE_DWORD
7 = enum_VAR_TYPE_FLOAT
8 = enum_VAR_TYPE_DOUBLE
9 = enum_VAR_TYPE_STRING
10 = enum_VAR_TYPE_ARRAY
11 = enum_VAR_TYPE_STRUCT
-2147220992 = enum_VAR_TYPE_E_UNKNOWN

Parameter	Description
None	None

Result enum eVariableType

Example:

```
Sub Main
        Dim objRet As DBVarObjCmdTarget
        Set objRet = GetVariableObject("VAR00001")
        MsgBox CStr(objRet.GetType()), vbOkOnly, "Test GetType"
        Set objRet = Nothing
End Sub
```

GetXMLSettings, DBVarObjCmdTarget Function

Syntax GetXMLSettings

Description This function returns the settings string of the variable in the project in XML format.

Parameter	Description
None	None

Result String

Example:

```
Sub Main
        Dim objRet As DBVarObjCmdTarget
        Set objRet = GetVariableObject("VAR00001")
        MsgBox CStr(objRet.GetXMLSettings()), vbOkOnly, GetProjectTitle
        Set objRet = Nothing
End Sub
```

IsOPCServerEnabled, DBVarObjCmdTarget Function

Syntax IsOPCServerEnabled

Description This function returns the True boolean result when the OPC server is enabled both in the Real Time DB and in the Option variable properties.

This function is not supported Windows CE.(If set, always returns 'false')

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Main
        Dim objRet As DBVarObjCmdTarget
        Set objRet = GetVariableObject("VAR00001")
        MsgBox CStr(objRet.IsOPCServerEnabled()), vbOkOnly, "Test IsOPCServerEnabled"
        Set objRet = Nothing
End Sub
```

IsOPCServerOnRequest, DBVarObjCmdTarget Function

Syntax IsOPCServerOnRequest

Description This function returns the True boolean result when the variable is listed in the OPC Server's items and when Clients are connected to it.

This function is not supported in Windows CE.(If set, always returns 'false')

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Main
        Dim objRet As DBVarObjCmdTarget
        Set objRet = GetVariableObject("VAR00001")
        MsgBox CStr(objRet.IsOPCServerOnRequest()), vbOkOnly, GetProjectTitle
        Set objRet = Nothing
End Sub
```

IsShared, DBVarObjCmdTarget Function

Syntax IsOPCServerEnabled

Description This function returns the True boolean when the variable's memory area has been set as shared (Input, Output, Flag).

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Main
        Dim objRet As DBVarObjCmdTarget
        Set objRet = GetVariableObject("VAR00001")
        MsgBox CStr(objRet.IsShared()), vbOkOnly, GetProjectTitle
        Set objRet = Nothing
End Sub
```

IsValid, DBVarObjCmdTarget Function

Syntax IsValid

Description This function returns the True boolean when the variable effectively has a valid value.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Main
        Dim objRet As DBVarObjCmdTarget
        Set objRet = GetVariableObject("VAR00001")
        MsgBox CStr(objRet.IsValid()), vbOkOnly, GetProjectTitle
        Set objRet = Nothing
End Sub
```

ResetStatisticData, DBVarObjCmdTarget Function

Syntax ResetStatisticData

Description This function allows you to reset the variable's statistic data. All statistic data value will be recalculated after the reset command.

Parameter	Description
None	None

Result None

Example:

```
Sub Main
        Dim objRet As DBVarObjCmdTarget
        Set objRet = GetVariableObject("VAR00001")
        objRet.ResetStatisticData()
        Set objRet = Nothing
End Sub
```

SetOPCServerEnabled, DBVarObjCmdTarget Function

Syntax SetOPCServerEnabled(_newVal)

Description This function lets you choose where to publish a variable in the OPC server DA and/or XML address space. The return value indicates that operation was successful and the variable will be published in the OPC Server. This function returns 'False' when executed while the OPC Server DA or XML was already being run, as a consequence this function can only be used at project startup path and therefore only in the basic script set as the startup script.

The startup basic script is executed in synchro. mode in respect to the started up project sources (even with 'Separate Thread' option active) with maximum timeout equal to the one set in the basic script resource's properties. As a consequence, it may be necessary to increase this value when using this function in a script

requiring more processing time before recommencing with starting up other remaining Movicon resources.

Parameter	Description
newVal As Boolean	True : Variable will be published in the opc server's address space. False : Variable will not be published in the opc server's address space.

Result Boolean

Example:

```
Option Explicit
Sub Main
	Dim objVar As DBVarObjCmdTarget
	Set objVar = CreateNewVar("VAR00002", enum_VAR_TYPE_DWORD, "") 'create variable
	If Not objVar Is Nothing Then
		If objVar.IsValid Then
			objVar.DynamicSettings = "[DRV]Modbus TCPIP.Sta=Default Station|Unit=1|FC=2|SA=0"
			objVar.SetOPCServerEnabled(True)
		Else
			Debug.Print Replace("The variable '%s' is not valid!", "%s", objVar.GetName)
		End If
	Else
		Debug.Print "Failed to create the variable 'RuntimeVariable'"
	End If
End Sub
```

SetTimeStamp, DBVarObjCmdTarget Function

Syntax SetTimeStamp()

Description This function allows you to set the variable's TimeStamp della variable with the system's current date.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
   Dim objVar As DBVarObjCmdTarget
   Set objVar = GetVariableObject("VAR00001")
   objVar.SetTimeStamp()
   MsgBox "Variable TimeStamp = " & CStr(objVar.GetTimeStamp()), vbInformation, GetProjectTitle
   Set objVar = Nothing
End Sub
```

AviFileProp, DBVarObjCmdTarget Property

Syntax AviFileProp = _String

Description This property allows you to get or set the string associated to the variable's "File Avi" property. This property can be interpreted by the connected OPC Client if predisposed with the necessary functionalities.

Parameter	Description
None	None

Result String

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget
        Set objVar = GetVariableObject("VAR00001")
        MsgBox "Avi file Prop = " & objVar.AviFileProp,vbInformation,GetProjectTitle
        Set objVar = Nothing
End Sub
```

BGColorProp, DBVarObjCmdTarget Property

Syntax BGColorProp = _Long

Description This property allows you to get or set the initial background color for the selected variable. This property can be interpreted by the OPC Client connected if provided with the right functions.

Parameter	Description
None	None

Result Long

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget
        Set objVar = GetVariableObject("VAR00001")
        MsgBox "BGColor Prop = " & objVar.BGColorProp,vbInformation,GetProjectTitle
        Set objVar = Nothing
End Sub
```

BlinkProp, DBVarObjCmdTarget Property

Syntax TRACEAddMsgLog = _Boolean

Description This property allows you to get or set the initial blink status. This property can be interpreted by the OPC Client connected if provided with the right functions.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget
        Set objVar = GetVariableObject("VAR00001")
        MsgBox "Blink Prop = " & objVar.BlinkProp,vbInformation,GetProjectTitle
        Set objVar = Nothing
End Sub
```

BmpFileProp, DBVarObjCmdTarget Property

Syntax BmpFileProp = _String

Description This property allows you to get or set the string associated to the "File Bitmap" property of a variable. This property can be interpreted by the OPC Client connected if provided with the right functions.

Parameter	Description
None	None

Result String

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget
        Set objVar = GetVariableObject("VAR00001")
        MsgBox "Bitmap file Prop = " & objVar.BmpFileProp,vbInformation,GetProjectTitle
        Set objVar = Nothing
End Sub
```

CloseBitString, DBVarObjCmdTarget Property

Syntax CloseBitString = _String

Description This property allows you to get or set the string associated to the "Close contact string" property of a variable. The string is actually associated to the variable's logic status "1" in runtime.

Parameter	Description

None	None

Result String

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget

        Set objVar = GetVariableObject("VAR00001")
        objVar.CloseBitString = "CLOSE"
        Set objVar = Nothing
End Sub
```

DynamicSettings, DBVarObjCmdTarget Property

Syntax DynamicSettings = _String

Description This property allows you to get or set the dynamic connection string for the specified variable. Corresponds to the "Dynamic Address" of a variable.

Parameter	Description
None	None

Result String

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget

        Set objVar = GetVariableObject("VAR00001")
        objVar.DynamicSettings = "[DRV]Modbus TCPIP.Sta=Station1|Unit=1|FC=2|SA=100"
        MsgBox "Dynamic Settings = " & objVar.DynamicSettings,vbInformation,GetProjectTitle
        Set objVar = Nothing
End Sub
```

EnableFactor, DBVarObjCmdTarget Property

Syntax EnableFactor = _Boolean

Description This property allows you to get or set the engineering data function of a variable connected to the Communication Driver. When enabled the values read from the field are written and scaled directly on the variable. The scaling is based on the settings from the FactorGain and FactorOffset properties.

Parameter	Description

None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim objVar As DBVarObjCmdTarget
        Set objVar = GetVariableObject("VAR00001")
        MsgBox("EnableFactor is: " & objVar.EnableFactor,vbOkOnly,GetProjectTitle)
        Set objVar = Nothing
End Sub
```

EnableNetworkServer, DBVarObjCmdTarget Property

Syntax EnableNetworkServer= _Boolean

Description When this property is enabled the variable can be shared with other Movicon applications through the Networking functions.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim objVar As DBVarObjCmdTarget
        Set objVar = GetVariableObject("VAR00001")
        MsgBox("EnableScalingFactor is: " &
        objVar.EnableNetworkServer,vbOkOnly,GetProjectTitle)
        Set objVar = Nothing
End Sub
```

EnableScalingFactor, DBVarObjCmdTarget Property

Syntax EnableScalingFactor = _Boolean

Description This property allows you to get or set the Engineering Data function of a variable connected to the Communication Driver. When enabled the values read from the field are written and scaled directly on the variable. The scaling is based on the settings from the 'Max. Scaled Value', 'Min. Scaled Value' , Max. Non Scaled Value' and 'Min. Non Scaled Value' properties.

Parameter	Description

None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim objVar As DBVarObjCmdTarget
        Set objVar = GetVariableObject("VAR00001")
        MsgBox("EnableScalingFactor is: " &
        objVar.EnableScalingFactor,vbOkOnly,GetProjectTitle)
        Set objVar = Nothing
End Sub
```

EngineeringUnit, DBVarObjCmdTarget Property

Syntax EngineeringUnit = _String

Description This property allows you to set or get the string associated to the "Engineering Unit" property of a variable. This actually represents the measure units to be displayed together with the variable's value.

Parameter	Description
None	None

Result String

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget

        Set objVar = GetVariableObject("VAR00001")
        objVar.EngineeringUnit = "mm"
        Set objVar = Nothing
End Sub
```

FactorGain, DBVarObjCmdTarget Property

Syntax FactorGain = _Double

Description This property allows you to get or set the gain value for variable scaling.

Parameter	Description
None	None

Result Double

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget

        Set objVar = GetVariableObject("VAR00001")
        objVar.FactorGain = 10
        Set objVar = Nothing
End Sub
```

FactorOffset, DBVarObjCmdTarget Property

Syntax FactorOffset = _Double

Description This property allows you to get or set the offset value for the variable scaling.

Parameter	Description
None	None

Result Double

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget

        Set objVar = GetVariableObject("VAR00001")
        objVar.FactorOffset = 5
        Set objVar = Nothing
End Sub
```

FGColorProp, DBVarObjCmdTarget Property

Syntax FGColorProp = _Long

Description This property allows you to get or set the foreground color for the selected variable. This property can be interpreted by the connected OPC Client if provided with the right functions.

Parameter	Description
None	None

Result Long

Example:

```
Sub Main
```

```
        Dim objVar As DBVarObjCmdTarget
        Set objVar = GetVariableObject("VAR00001")
        MsgBox "FGColor Prop = " & objVar.FGColorProp,vbInformation,GetProjectTitle
        Set objVar = Nothing
End Sub
```

Group, DBVarObjCmdTarget Property

Syntax Group = _String

Description This property returns any variable group belonging to the reference variable (Folder where variable belongs).

Parameter	Description
None	None

Result String

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget

        Set objVar = GetVariableObject("VAR00001")
        MsgBox "Group = " & objVar.Group,vbInformation,GetProjectTitle
End Sub
```

HtmlFileProp DBVarObjCmdTarget Property

Syntax HtmlFileProp = _String

Description This property allows you to get or set the string associated to the "File Html" of a variable. Once set it can be interpreted by the OPC Client connected if provided with right functions.

Parameter	Description
None	None

Result String

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget
        Set objVar = GetVariableObject("VAR00001")
        MsgBox "Html file Prop = " & objVar.HtmlFileProp,vbInformation,GetProjectTitle
        Set objVar = Nothing
End Sub
```

InUse, DBVarObjCmdTarget Property

Syntax InUse = _Boolean

Description This property allows you to get or set the "In Use" status of the specified variable. The forcing of the variable's In Use status is managed by the communication driver or the OPC etc. For instance by forcing the property to "True" the variable will result as being in use and therefore it will be updated by the communication driver. On the other hand, when forcing the property to "False" the variable will result as being not in use and will be kept updated by the communication driver according to the refresh times for variables not in use.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget

        Set objVar = GetVariableObject("VAR00001")
        MsgBox "In Use = " & objVar.InUse,vbInformation,GetProjectTitle
End Sub
```

InverseFactor, DBVarObjCmdTarget Property

Syntax InverseFactor = _Boolean

Description When this property is enabled the variable's scaling is done with inverse factors.

When the InverseFactor assumes the True Boolean value the instruction will be:
(Value - Offset)/Gain

When the InverseFactor assumes the False Boolean value the instruction will be:
(Value *Gain)+ Offset

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim objVar As DBVarObjCmdTarget
        Set objVar = GetVariableObject("VAR00001")
        MsgBox("EnableScalingFactor is: " & objVar.InverseFactor,vbOkOnly,GetProjectTitle)
        Set objVar = Nothing
```

```
End Sub
```

InverseScaling, DBVarObjCmdTarget Property

Syntax InverseScaling = _Boolean

Description When this property is enabled the variable is scaled in the inverse to the one set. Let's take the following settings as an example:

Not Scaled Max. Value = 100
Not Scaled Min. Value = 0
Scaled Max. Value = 1000
Scaled Min. Value = 0

When the variable obtains the 0 real value, the scaled value will be 1000 and when the variable obtains the 100 scaled value, the scaled value will be 0.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim objVar As DBVarObjCmdTarget
        Set objVar = GetVariableObject("VAR00001")
        MsgBox("EnableScalingFactor is: " & objVar.InverseScaling ,vbOkOnly,GetProjectTitle)
        Set objVar = Nothing
End Sub
```

LastChangeComment, DBVarObjCmdTarget Property

Syntax LastChangeComment = _String

Description This property returns the last comment inserted on variable change. This property has effect when the 'Trace Comment' property has been enabled.

Parameter	Description
None	None

Result String

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget
```

```
        Set objVar = GetVariableObject("VAR00001")
        MsgBox "Last Change Comment = " &
        objVar.LastChangeComment,vbInformation,GetProjectTitle
End Sub
```

MapRealTimeODBCUpdateQuality, DBVarObjCmdTarget Property

Syntax MapRealTimeODBCUpdateQuality = _Boolean

Description This property allows you to get or set the Update Quality function in the Real Time ODBC (Property window) for the specified variable.

Parameter	Description
lpar As Boolean	None

Result Long

Example:

```
Sub Main
        Dim objRet As DBVarObjCmdTarget
        Dim result As Boolean
        Set objRet = GetVariableObject("VAR00001")
        result = objRet.MapRealTimeODBCUpdateQuality ()
        MsgBox "MapRealTimeODBCUpdateQuality  = " & result ,vbOkOnly, GetProjectTitle
        Set objRet = Nothing
End Sub
```

MapRealTimeToDB, DBVarObjCmdTarget Property

Syntax MapRealTimeToDB = _Boolean

Description This property allows you to get or set the sharing function with the Database, enabled for this purpose, for the specified variable. By doing this the variable will be made available on a Database and therefore also accessible from other applications. Corresponds to the "Enable" property of the "ODBC Real Time Properties' of a variable.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget
```

```
        Set objVar = GetVariableObject("VAR00001")
        MsgBox "Map RealTime To DB = " &
        objVar.MapRealTimeToDB,vbInformation,GetProjectTitle
End Sub
```

MapRealTimeToDBMode, DBVarObjCmdTarget Property

Syntax MapRealTimeToDBMode = _Byte

Description This property allows you to get or set the mode with which the specified variable is to be exchanged with the Database enabled for this purpose. Corresponds to the "Mode" property from the variable Real Time ODBC Properties.

The values which can be set or returned are:

0 = Input
1 = Output
2 = Input/Output

Parameter	Description
None	None

Result Byte

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget

        Set objVar = GetVariableObject("VAR00001")
        MsgBox "Map RealTime To DB Mode = " &
        objVar.MapRealTimeToDBMode,vbInformation,GetProjectTitle
End Sub
```

MapRealTimeToDBRefreshTime, DBVarObjCmdTarget Property

Syntax MapRealTimeToDBRefreshTime = _Long

Description This property allows you to set or get the Refresh Time with which the variables are read and written from the associated Database. Corresponds to the "Reading Refresh Time" properties from the variable's "Real Time ODBC properties".

Parameter	Description
None	None

Result Long

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget

        Set objVar = GetVariableObject("VAR00001")
        MsgBox "Map RealTime To DB Refresh Time = " &
        objVar.MapRealTimeToDBRefreshTime,vbInformation,GetProjectTitle
End Sub
```

NetworkClientEnable, DBVarObjCmdTarget Property

Syntax NetworkClientEnable = _Boolean

Description This property allows you enable the Networking connection between the variable and the local project and any other Movicon station setup as Server.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget

        Set objVar = GetVariableObject("VAR00001")
        MsgBox "Network Client Enable = " &
        objVar.NetworkClientEnable,vbInformation,GetProjectTitle
End Sub
```

NetworkClientMode, DBVarObjCmdTarget Property

Syntax NetworkClientMode = _Byte

Description This propery allows you to get or set the mode with which the specified variable will be exhanged with the Server in network. Corresponds to the "Mode" property from the variable's "Network Client Properties".

The setting and return values are:

0 = Input
1 = Output
2 = Input/Output

Parameter	Description
None	None

Result Byte

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget

        Set objVar = GetVariableObject("VAR00001")
        MsgBox "Network Client Mode = " &
        objVar.NetworkClientMode,vbInformation,GetProjectTitle
End Sub
```

NetworkClientServerName, DBVarObjCmdTarget Property

Syntax NetworkClientServerName = _String

Description This property allows you to get or set the name of the Network Server to which the specified variable is to connect. Corresponds to the "Network Server" from the variable's "Network Client Properties".

Parameter	Description
None	None

Result String

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget

        Set objVar = GetVariableObject("VAR00001")
        MsgBox "Network Client Server Name = " &
        objVar.NetworkClientServerName,vbInformation,GetProjectTitle
End Sub
```

NetworkClientUpdateQuality, DBVarObjCmdTarget Property

Syntax NetworkClientUpdateQuality = _Boolean

Description This property allows you to get or read the Update Quality function in the Network Client section (Property window) for the specified window.

Parameter	Description
lpar As Boolean	None

Result Long

Example:

```
Sub Main
        Dim objRet As DBVarObjCmdTarget
        Dim result As Boolean
        Set objRet = GetVariableObject("VAR00001")
        result = objRet.NetworkClientUpdateQuality ()
        MsgBox "NetworkClientUpdateQuality = " & result ,vbOkOnly, GetProjectTitle
        Set objRet = Nothing
End Sub
```

NetworkServerEnable, DBVarObjCmdTarget Property

Syntax NetworkServerEnable = _Boolean

Description This property allows you to set or get the Network Server function for the specified variable. This will make the variable available to any network Client connections. Corresponds to the "Enable Network Server" property of the variable's "Options Properties".

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget

        Set objVar = GetVariableObject("VAR00001")
        MsgBox "Network Server Enable = " &
        objVar.NetworkServerEnable,vbInformation,GetProjectTitle
End Sub
```

NetworkServerIsWritable, DBVarObjCmdTarget Property

Syntax NetworkServerIsWritable = _Boolean

Description This property allows you to get or set the variable's write function on the specified Server for Client connections.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget

        Set objVar = GetVariableObject("VAR00001")
        MsgBox "Network Server Is Writable = " &
        objVar.NetworkServerIsWritable,vbInformation,GetProjectTitle
End Sub
```

OPCGroupName, DBVarObjCmdTarget Property

Syntax OPCGroupName = _String

Description This property allows you to read or set the name of the OPC group (in the project structure) to which the reference variable belongs.

Parameter	Description
None	None

Result String

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget

        Set objVar = GetVariableObject("VAR00001")
        Debug.Print objVar.OPCGroupName
End Sub
```

OPCServerAccessRights,DBVarObjTarget Property

Syntax EnableScalingFactor = eOPCServerMode

Description This property allows you to select access type to Movicon OPC Server items for OPC Client side.
Values allowed can be specified using the eOPCServerMode enumerator or by inserting the corresponding numeric values:

enum_opc_readable (vaue 1, readable)
enum_opc_writeable (value 2, writable)

enum_opc_readable_writeable (Value 3, readable-writeable)

In cases where the access type from "readable-writable" to "readable" is set, an OPC Client will only be able to read the value, but will still be published as read and write in the item's property.

Parameter	Description
None	None

Result eOPCServerMode

Example:

```
Sub Main
        Dim var1 As DBVarObjCmdTarget
        Set var1=GetVariableObject("VAR00001")
        var1.OPCServerAccessRights = enum_opc_readable
        Set var1=Nothing
End Sub
```

OpenBitString, DBVarObjCmdTarget Property

Syntax OpenBitString = _String

Description This property allows you to set or get the string associated to the variable's "Open Bit String" property. This is the string which will be associated to the variable's "0" logic status in runtime.

Parameter	Description
None	None

Result String

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget

        Set objVar = GetVariableObject("VAR00001")
        objVar.OpenBitString = "OPEN"
End Sub
```

Quality, DBVarObjCmdTarget Property

Syntax Quality = _Integer

Description This property allows you to get the quality status of the specified variable. The property is in read only.

The returned values tally with the OPC specification quality values:

252 = OPC_STATUS_MASK
3 = OPC_LIMIT_MASK
0 = OPC_QUALITY_BAD
64 = OPC_QUALITY_UNCERTAIN
192 = OPC_QUALITY_GOOD
4 = OPC_QUALITY_CONFIG_ERROR
8 = OPC_QUALITY_NOT_CONNECTED
12 = OPC_QUALITY_DEVICE_FAILURE
16 = OPC_QUALITY_SENSOR_FAILURE
20 = OPC_QUALITY_LAST_KNOWN
24 = OPC_QUALITY_COMM_FAILURE
28 = OPC_QUALITY_OUT_OF_SERVICE
68 = OPC_QUALITY_LAST_USABLE
80 = OPC_QUALITY_SENSOR_CAL
84 = OPC_QUALITY_EGU_EXCEEDED
88 = OPC_QUALITY_SUB_NORMAL
216 = OPC_QUALITY_LOCAL_OVERRIDE

Parameter	Description
None	None

Result Integer

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget

        Set objVar = GetVariableObject("VAR00001")
        MsgBox "VAR00001 = " & objVar.Quality,vbInformation,GetProjectTitle
End Sub
```

ScaleMax, DBVarObjCmdTarget Property

Syntax ScaleMax = _Double

Description This property allows you to get or set the maximum scaled value of the specified variable. Corresponds to the variable's "Scale Max." property.

Parameter	Description
None	None

Result Double

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget

        Set objVar = GetVariableObject("VAR00001")
        MsgBox "Scale Max = " & objVar.ScaleMax,vbInformation,GetProjectTitle
        Set objVar = Nothing
End Sub
```

ScaleMin, DBVarObjCmdTarget Property

Syntax ScaleMin = _Double

Description This property allows you to get or set the minimum scaled value of the specified variable. Corresponds to the variable's "Scale Min." property.

Parameter	Description
None	None

Result Double

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget

        Set objVar = GetVariableObject("VAR00001")
        MsgBox "Min = " & objVar.ScaleMin,vbInformation,GetProjectTitle
        Set objVar = Nothing
End Sub
```

ScaleRawMax, DBVarObjCmdTarget Property

Syntax ScaleRawMax = _Double

Description This property allows you to get or set the maximum raw value of the specified variable. Corresponds to the variable's "Raw Max." property.

Parameter	Description
None	None

Result Double

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget

        Set objVar = GetVariableObject("VAR00001")
        MsgBox "Scale Raw Max = " & objVar.ScaleRawMax,vbInformation,GetProjectTitle
        Set objVar = Nothing
End Sub
```

ScaleRawMin, DBVarObjCmdTarget Property

Syntax ScaleRawMin = _Double

Description This property allows you to get or set the minimum raw value of the specified variable. Corresponds to the variable's "Raw Min." property.

Parameter	Description
None	None

Result Double

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget

        Set objVar = GetVariableObject("VAR00001")
        MsgBox "Scale Raw Min = " & objVar.ScaleRawMin,vbInformation,GetProjectTitle
        Set objVar = Nothing
End Sub
```

SndFileProp DBVarObjCmdTarget Property

Syntax SndFileProp = _String

Description This property allows you to get or set the string associated to the variable's "Sound File" property. The associated sound file has to be .wav type. Once set, it can be interpreted by the connected OPC client if provided with the right functions.

Parameter	Description
None	None

Result String

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget
        Set objVar = GetVariableObject("VAR00001")
        MsgBox "Sound file Prop = " & objVar.SndFileProp,vbInformation,GetProjectTitle
        Set objVar = Nothing
End Sub
```

StatisticData DBVarObjCmdTarget Property

Syntax StatisticData = _Boolean

Description This property is read only and lets you know whether the statistic data for the variable in question has been enabled or not. In this case it will possible to get the variable's minimum, maximum and average values and other information.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget
        Set objVar = GetVariableObject("VAR00001")
        MsgBox "Enable Statistic Data = " & objVar.StatisticData,vbInformation,GetProjectTitle
        Set objVar = Nothing
End Sub
```

StatisticDataAverage, DBVarObjCmdTarget Property

Syntax StatisticDataAverage = _Variant

Description This property allows you to set or get the variable's statistic data average. In order to use this property you need to set the Statistic Data property to TRUE. This property is read only.

Parameter	Description
None	None

Result Variant

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget
        Set objVar = GetVariableObject("VAR00001")
        MsgBox "Statistic Data Average Value = " &
        objVar.StatisticDataAverage,vbInformation,GetProjectTitle
        Set objVar = Nothing
End Sub
```

StatisticDataMaxValue, DBVarObjCmdTarget Property

Syntax StatisticDataMaxValue = _Variant

Description This property allows you to get the variable's statistic data maximum value. In order to use this property you need to set the StatisticData property to TRUE. This property is in read only.

Parameter	Description
None	None

Result Variant

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget
        Set objVar = GetVariableObject("VAR00001")
        MsgBox "Statistic Data Max Value = " &
        objVar.StatisticDataMaxValue,vbInformation,GetProjectTitle
        Set objVar = Nothing
End Sub
```

StatisticDataMinValue, DBVarObjCmdTarget Property

Syntax StatisticDataMinValue = _Variant

Description This property allows you to get the variable's statistic data minimum value. In order to use this property you need to set the StatisticData property to TRUE. This property is read only.

Parameter	Description
None	None

Result Variant

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget
        Set objVar = GetVariableObject("VAR00001")
        MsgBox "Statistic Data Min Value = " &
        objVar.StatisticDataMinValue,vbInformation,GetProjectTitle
        Set objVar = Nothing
End Sub
```

StatisticDataNumSamples, DBVarObjCmdTarget Property

Syntax StatisticDataNumSamples = _Long

Description This property allows you to get the number of samples performed on a variable. In order to use this property you need to set the StatisticData property to TRUE. This property is read only.

Parameter	Description
None	None

Result Long

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget
        Set objVar = GetVariableObject("VAR00001")
        MsgBox "Statistic Data Num. Samples = " &
        objVar.StatisticDataNumSamples,vbInformation,GetProjectTitle
        Set objVar = Nothing
End Sub
```

StatisticTotalTimeOn, DBVarObjCmdTarget Property

Syntax StatisticTotalTimeOn

Description This property returns double type value indicating the total time in seconds in which the variable has a value different from zereo.
This data is saved in the variable's retentive file.
This property is read only.

Parameter	Description
None	None

Result Double

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget
        Set objVar = GetRealTimeDB.GetVariableObject("Alarm01")
        MsgBox " TotalTimeOn = " & CStr(objVar.StatisticTotalTimeOn)
        Set objVar = Nothing
End Sub
```

TraceAddDescCol, DBVarObjCmdTarget Property

Syntax TraceAddDescCol = _Boolean

Description This property consents you to enable or disable the recording of the Variable Description in the Trace table for the variable specified. This corresponds to the Variable "Add Variable Description Column" property.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget

        Set objVar = GetVariableObject("VAR00001")
        MsgBox "Trace Add Tag Description= " &
        objVar.TraceAddDescCol,vbInformation,GetProjectTitle
End Sub
```

TraceAddGroupCol, DBVarObjCmdTarget Property

Syntax TraceAddGroupCol = _Boolean

Description This property consents you to enable or disable the recording of the Variable Group in the Trace table. This corresponds to the Variable "Add Variable Group Column" property.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget

        Set objVar = GetVariableObject("VAR00001")
        MsgBox "Trace Add Tag Group= " &
        objVar.TraceAddGroupCol,vbInformation,GetProjectTitle
End Sub
```

TraceAddMsgLog, DBVarObjCmdTarget Property

Syntax TraceAddMsgLog = _Boolean

Description This property allows you to get or set the recording function in the "SysLog" (Historical Log of System Messages) of messages each time the specified variable changes value. Corresponds to the variable's "Add Msg to SysLog" property.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget

        Set objVar = GetVariableObject("VAR00001")
        MsgBox "Trace Add Msg Log = " & objVar.TraceAddMsgLog,vbInformation,GetProjectTitle
End Sub
```

TraceComment, DBVarObjCmdTarget Property

Syntax TraceComment = _Boolean

Description This property, when set with the True boolean value, allows you to insert a comment about the variable in the trace Database every time it changes. This command will be recorded in the "Action" field replacing the text Movicon would have written for default. For further information please refer toVariable Trace Options Properties.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget

        Set objVar = GetVariableObject("VAR00001")
        MsgBox "Trace Comment = " & objVar.TraceComment,vbInformation,GetProjectTitle
End Sub
```

TraceEnable, DBVarObjCmdTarget Property

Syntax TraceEnable = _Boolean

Description This property allows you to get or set the trace functions for the specified variable. Corresponds to the variable's 'Enable Trace' property.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget

        Set objVar = GetVariableObject("VAR00001")
        MsgBox "Trace Enable = " & objVar.TraceEnable,vbInformation,GetProjectTitle
End Sub
```

TraceEnableFromToTime, DBVarObjCmdTarget Property

Syntax TRACEEnableFromToTime = _Boolean

Description This property allows you to get or set the daily timeframe for the trace recording for the specified variable. Corresponds to the variable's "Enable Day Timeframe" property. When this property is enabled the trace recordings of the variable will be done only withing the time specified in the "TimeFrame From" and "TimeFrame To" properties.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget

        Set objVar = GetVariableObject("VAR00001")
        MsgBox "TRACE Enable From To Time = " &
        objVar.TRACEEnableFromToTime,vbInformation,GetProjectTitle
End Sub
```

TraceFromTime, DBVarObjCmdTarget Property

Syntax TraceFromTime = _Date

Description This property allows you to set or get the start time of the specified variable's tracing. Corresponds to the variable's "TimeFrame From" property. This property can only go into effect when the "Enable Day

TimeFrame" has been enabled.

Parameter	Description
None	None

Result Date

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget

        Set objVar = GetVariableObject("VAR00001")
        MsgBox "Trace From Time = " & objVar.TraceFromTime,vbInformation,GetProjectTitle
End Sub
```

TraceMaxAgeDays, DBVarObjCmdTarget Property

Syntax TraceMaxAgeDays = _Long

Description This property allows you to get or set the maximum time expressed in days after which the specified varable's traced data will begin to recycle. Corresponds to the Variable's "Data MaxAge" property.
The "Data MaxAge" property is calculated on the settings of the "TraceMaxAgeDays", "TraceMaxAgeHours" and "TraceMaxAgeMins" properties.

Parameter	Description
None	None

Result Long

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget

        Set objVar = GetVariableObject("VAR00001")
        MsgBox "Trace Max Age Days = " &
        objVar.TraceMaxAgeDays,vbInformation,GetProjectTitle
End Sub
```

TraceMaxAgeHours, DBVarObjCmdTarget Property

Syntax TraceMaxAgeHours = _Long

Description This property allows you to get or set the maximum time expressed in hours after which the variable's traced data begins to be recycled. Corresponds to the variable's "Data MaxAge" property.
The "Data MaxAge" property is calculated on the settings of the "TraceMaxAgeDays", "TraceMaxAgeHours" and "TraceMaxAgeMins" properties.

Parameter	Description
None	None

Result Long

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget

        Set objVar = GetVariableObject("VAR00001")
        MsgBox "Trace Max Age Hours = " &
        objVar.TraceMaxAgeHours,vbInformation,GetProjectTitle
End Sub
```

TraceMaxAgeMins, DBVarObjCmdTarget Property

Syntax TraceMaxAgeMins = _Long

Description This property allows you to get or set the maximum time expressed in minutes after which the variable's traced data will begin to be recycled. Corresponds to the variable's "Data MaxAge" property.
The "Data MaxAge" property is calculated on the settings of the "TraceMaxAgeDays", "TraceMaxAgeHours" and "TraceMaxAgeMins" properties.

Parameter	Description
None	None

Result Long

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget

        Set objVar = GetVariableObject("VAR00001")
        MsgBox "Trace Max Age Minutes= " &
        objVar.TraceMaxAgeMins,vbInformation,GetProjectTitle
End Sub
```

TraceTableName, DBVarObjCmdTarget Property

Syntax TraceTableName = _String

Description This property allows you to set or get the name of the Trace Table for the specified variable. Corresponds to the variable's "Table Name" property.

Parameter	Description
None	None

Result String

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget

        Set objVar = GetVariableObject("VAR00001")
        MsgBox "Trace Table Name = " & objVar.TraceTableName,vbInformation,GetProjectTitle
End Sub
```

TraceToTime, DBVarObjCmdTarget Property

Syntax TraceToTime = _Date

Description This property allows you to get or set the time in which the variable's tracing is to end. Corresponds to the variable's "TimeFrame To" property. This property can only go into effect when the "Enable Day TimeFrame" has been enabled.

Parameter	Description
None	None

Result Date

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget

        Set objVar = GetVariableObject("VAR00001")
        MsgBox "Trace to Time = " & objVar.TraceToTime,vbInformation,GetProjectTitle
End Sub
```

Value, DBVarObjCmdTarget Property

Syntax Value = _Variant

Description This property allows you to read or set the value of a variable.

Parameter	Description
None	None

Result Variant

Example:

```
Sub Main
        Dim objVar As DBVarObjCmdTarget

        Set objVar = GetVariableObject("VAR00001")
        MsgBox "VAR00001 = " & objVar.Value,vbInformation,GetProjectTitle
End Sub
```

25. DisplayEditCmdTarget

GetComboListInterface, DisplayEditCmdTarget Function

Syntax GetComboListInterface()

Description This property returns the combobox's list object. The methods and properties are those specified in the ListBoxCmdTarget interface.

Parameter	Description
None	None

Result Object
If Function has been executed successfully it will retrieve an object of type ListBoxCmdTarget if otherwise Nothing is returned.

Example:

```
Public Sub Click()
        Dim obj As ListBoxCmdTarget
        Set obj = GetComboListInterface
        obj.AddString("prova!!")
End Sub
```

IsCombo, DisplayEditCmdTarget Function

Syntax IsCombo()

Description This property returns True when the display belongs to a Movicon Combobox object.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dbug.Print IsCombo
End Sub
```

IsSpin, DisplayEditCmdTarget Function

Syntax IsSpin()

Description This property returns True when the object is a Movicon Spin.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dbug.Print IsSpin
End Sub
```

LoadExtSettings, DisplayEditCmdTarget Function

Syntax LoadExtSettings()

Description This function permits you to load the object's settings from the releative external settings file. This file can be specified in in the "Settings File" property in design mode, or by using the "ExtSettingsFile" interface property. The extension provided for this file is ".SXML".

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim objSymbol As LoadExtSettings, DisplayEditCmdTarget Function
        Set objSymbol =
        GetSynopticObject.GetSubObject("TestObject").GetObjectInterface
        If objSymbol Is Nothing Then Exit Sub
        objSymbol.ExtSettingsFile = "test.sxml"
        objSymbol.LoadExtSettings
        Set objSymbol = Nothing
End Sub
```

RecalcLayout, DisplayEditCmdTarget Function

Syntax RecalcLayout()

Description This function updates the object's graphical layout according to the changes made.

Parameter	Description
None	None

Result None

Example:

```
Public Sub Click()
        If IsCombo Then
                GetObjectInterface.RecalcLayout
        End If
End Sub
```

SaveExtSettings, DisplatEditCmdTarget Function

Syntax SaveExtSettings

Description This function permits you to save the object's configurations in the relating external setting file. This file can be specified in design mode in the "Ext.File Settings property", or using the "ExtSettingsFile" interface property. This extension provided for this file is ".SXML".

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
Dim objSymbol As DisplayEditCmdTarget
Set objSymbol = GetSynopticObject.GetSubObject("TestObject").GetObjectInterface
If objSymbol Is Nothing Then Exit Sub
objSymbol.ExtSettingsFile = "test.sxml"
objSymbol.SaveExtSettings
Set objSymbol = Nothing
End Sub
```

EditingPassword, DisplayEditCmdTarget Property

Syntax EditingPassword = _Boolean

Description This property enables or disables the displaying the display's value in password format. Once this property has been modified, you will need to use the "RecalcLayout" function to apply modification made.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim objSyn As SynopticCmdTarget
        Dim objDisplay As DisplayEditCmdTarget

        Set objSyn = GetSynopticObject
        If Not objSyn Is Nothing Then
                Set objDisplay = objSyn.GetSubObject("MyDisplay").GetObjectInterface
                If Not objDisplay Is Nothing Then
                        objDisplay.EditingPassword = Not objDisplay.EditingPassword
                        objDisplay.RecalcLayout
                        Set objDisplay = Nothing
                End If
                Set objSyn = Nothing
        End If
End Sub
```

FormatData, DisplayEditCmdTarget Property

Syntax Format = _String

Description This property sets or returns a text for identifying the variable's format to be represented. Accepts a String type value.

The format types available are represented by the following syntax types:

x: where the x number identifies the number of figures to be displayed.
x.x: where the x number after the decimal point indicated the number of decimal figures to be displayed.

The format types with decimal figures, "x,x" have meaning when a variable in "floating point" format has been inserted. When "integer" numbers are used the decimal figures will always remain at zero.

Parameter	Description
None	None

Result String

Example:
```
Public Sub Click()
        Format = InputBox("Insert value")
End Sub
```

ExtSettingsFile, DisplayEditCmdTarget Property

Syntax ExtSettingsFile = _String

Description This property sets or returns the external configuration file for the referenced object. This file can also be specified in design mode in the object's 'Ext. File Settings' property. The extension provided for this file is ".SXML".

Parameter	Description
None	None

Result String

Example:
```
Public Sub Click()
Dim objSymbol As DisplayEditCmdTarget
Set objSymbol = GetSynopticObject.GetSubObject("TestObject").GetObjectInterface
If objSymbol Is Nothing Then Exit Sub
objSymbol.ExtSettingsFile = "test.sxml"
objSymbol.SaveExtSettings
Set objSymbol= Nothing
End Sub
```

FormatVariable, DisplayEditCmdTarget Property

Syntax FormatVariable = _String

Description This property resturns or sets the name of the variable whose value will be used for determining the display format of the variable associated to the Display.

Parameter	Description
None	None

Result String

Example:
```
Public Sub Click()
        Dim objDisplay As DisplayEditCmdTarget
```

```
        Set objDisplay =
        GetSynopticObject.GetSubObject("Display1").GetObjectInterface
        MsgBox  "Display1 Format Variable = " & objDisplay.FormatVariable,
        vbInformation,GetProjectTitle
        Set objDisplay = Nothing
End Sub
```

HasSpin, DisplayEditCmdTarget Property

Syntax HasSpin = _Boolean

Description This property enables or disables the possibility to associated the Spin Button to the display so that the variable's value can be increased or decreased without having to use the keyboard.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        HasSpin = True
End Sub
```

InvertSelection, DisplayEditCmdTarget Property

Syntax InvertSelection = _Boolean

Description This property is used for defining whether the combo-box list is to open towards the top or bottom. When setting this property with the "False" value the list eill open towards the bottom, when setting it to the "True" it will open towards the top. Once this property has been modified, it will be necessary to execute the "RecalcLayout" function to apply modification.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim objSyn As SynopticCmdTarget
        Dim objDipslay As DisplayEditCmdTarget

        Set objSyn = GetSynopticObject
        If Not objSyn Is Nothing Then
                Set objDipslay = objSyn.GetSubObject("MyDisplay").GetObjectInterface
                If Not objDipslay Is Nothing Then
```

```
                    objDipslay.InvertSelection = Not objDipslay.InvertSelection
                    objDipslay.RecalcLayout
                    Set objDipslay = Nothing
                End If
                Set objSyn = Nothing
        End If
End Sub
```

IsEditable, DisplayEditCmdTarget Property

Syntax IsEditable = _Boolean

Description When setting this property to False the display will become read only.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        IsEditable = True
End Sub
```

IsSpinHoriz, DisplayEditCmdTarget Property

Syntax IsSpinHoriz = _Boolean

Description This property allows you set to the Spin button to vertical or horizontal.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        IsSpinHoriz = True
End Sub
```

PromptPad, DisplayEditCmdTarget Property

Syntax LoadExtSettings

Description This property enables or disables the option to display the Numeric or Alphanumeric Pad when the user clicks on the editable display. The Numeric Pad appears if the variable is numeric type, otherwise the Alphanumeric pad will show.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim objDisplay As DisplayEditCmdTarget
        Set objDisplay = GetSynopticObject.GetSubObject("Display1").GetObjectInterface
        objDisplay.PromptPad = Not objDisplay.PromptPad
        Set objDisplay = Nothing
End Sub
```

ScaleUnit, DisplayEditCmdTarget Property

Syntax ScaleUnit = _String

Description This property sets or returns a text for identifying the variable measure units to be displayed. Accepts a Sting type value.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        ScaleUnit = InputBox("Insert value")
End Sub
```

SpinStep, DisplayEditCmdTarget Property

Syntax SpinStep = _Double

Description This property allows you to set the increment or decrement value which will be applied to the variable with the Spin buttons.

Parameter	Description
None	None

Result Double

Example:

```
Public Sub Click()
        Dim dRet As Double
        SpinStep = "VAR0001"
        dRet = SpinStep
        MsgBox "SpinStep = " & sRet, vbOkOnly, GetProjectTitle
End Sub
```

TimeToWaitToIncrease, DisplayEditCmdTarget Property

Syntax TimeToWaitToIncrease = _Long

Description This property returns or sets the delay time to the enabling of the fast Increase/Decrease function. This will make the variable begin to Increase/Decrease in fast mode upon the expiry of the time set when one of the spin buttons is kept pressed down with the mouse.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim sRet As Long
        TimeToWaitToIncrease = "VAR0001"
        sRet = ValMin
        MsgBox "ValMin = " & sRet, vbOkOnly, GetProjectTitle
End Sub
```

ValMax, DisplayEditCmdTarget Property

Syntax ValMax = _Double

Description This property returns or sets the maximum value which the Display's variable can obtain.

*The **"Val. Max."** does not have effect if a variable has been entered in the "Var. Max." property of the Display's "**Variable Properties**" for managing thresholds in dynamic mode.*

Parameter	Description
None	None

Result Double

Example:

```
Public Sub Click()
        Dim sRet As Double
        ValMax =250
        sRet = ValMax
        MsgBox "ValMax = " & sRet, vbOkOnly, GetProjectTitle
End Sub
```

ValMin, DisplayEditCmdTarget Property

Syntax ValMin = _Double

Description This property returns or sets the minimum value which the Display's variable can obtain.

*The **"Val. Min."** does not have effect if a variable has been entered in the "Var. Min." property of the Display's "**Variable Properties**" for managing thresholds in dynamic mode.*

Parameter	Description
None	None

Result Double

Example:

```
Public Sub Click()
        Dim sRet As Double
        ValMin = 0
        sRet = ValMin
        MsgBox "ValMin = " & sRet, vbOkOnly, GetProjectTitle
End Sub
```

ValueToDisplay, DisplayEditCmdTarget Property

Syntax ValueToDisplay = eEditDisplayStatistics

Description This property allows you to read or set the data type to display for the variable associated to the Display. The available value can be selected either by using the eEditDisplayStatistics enumerator or by inserting the numeric value:

enum_eds_none: actual value (value 0)
enum_eds_min: minimum value (value 1)
enum_eds_max: Maximum value (value 2)
enum_eds_average: average value (value 3)
enum_eds_TotalTime: Total time ON(value 4)
enum_eds_min_Day: minimum day value (value 5)
enum_eds_max_Day: maximum day value (value 6)
enum_eds_average_Day: average day value (valore 7)
enum_eds_TotalTime_Day: Total day time On (value 8)

enum_eds_min_Week: minimum week value (value 9)
enum_eds_max_Week: maximum week value (value 10)
enum_eds_average_Week: average week value (value 11)
enum_eds_TotalTime_Week: total week time ON (value 12)
enum_eds_min_Month: minimum month value (value 13)
enum_eds_max_Month: maximum month value (value 14)
enum_eds_average_Month: average month value (value 15)
enum_eds_TotalTime_Month: total month time ON (value 16)
enum_eds_min_Year: minimum year value (value 17)
enum_eds_max_Year: maximum year value (value 18)
enum_eds_average_Year: average year value (value 19)
enum_eds_TotalTime_Year: Total year time ON (value 20)
enum_eds_TimeStamp: late update time, TimeStamp (value 21)

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        MsgBox "Value Displaied = " & ValueToDisplay, vbOkOnly, GetProjectTitle
End Sub
```

Variable, DisplayEditCmdTarget Property

Syntax Variable = _String

Description This property returns or sets the name of the variable to be displayed or set by means of using the Display.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sRet As String
        Variable = "VAR0001"
        sRet = Variable
        MsgBox "Variable = " & sRet, vbOkOnly, GetProjectTitle
End Sub
```

VariableMax, DisplayEditCmdTarget Property

Syntax VariableMax = _String

Description This property returns or sets the name of the variable whose value is to be used as the maximum value which can be set in the "Variable". By doing this the threshold's value will become dynamic so it can also be changed during the Runtiime phase.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sRet As String
        VariableMax = "VAR0001"
        sRet = VariableMax
        MsgBox "VariableMin = " & sRet, vbOkOnly, GetProjectTitle
End Sub
```

VariableMin, DisplayEditCmdTarget Property

Syntax VariableMin = _String

Description This property returns or set the name of the variable whose value is to be used as the minimum value which canbe set in the "Variable". By doing this the threshold's value will become dynamic so it can also be changed during the Runtiime phase.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sRet As String
        VariableMin = "VAR0001"
        sRet = VariableMin
        MsgBox "VariableMin = " & sRet, vbOkOnly, GetProjectTitle
End Sub
```

26. DLRCmdTarget

GetADOConn, DLRCmdTarget Function

Syntax GetADOConn()

Description This function returns a ADODB Connection object for the connection to the datalogger by means of using script codes.

If used in Windows CE, this function will always return an ADOCE.connection.3.1. type object. Furthermore, avoid using the "close method" to close ADO connections, otherwise Movicon will no longer be able to access that database.

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        Dim Conn1 As New ADODB.Connection
        Dim Rs1 As New ADODB.Recordset
        Dim contFields As Integer
        Dim sQuery As String
        Set Conn1 = objDLR.GetADOConn
        sQuery = "SELECT * FROM Log5sec"
        Set Rs1 = CreateObject("ADODB.Recordset")
        Rs1.Open sQuery, Conn1, adOpenForwardOnly, adLockReadOnly, ADODB.adCmdText
        ' Loop per stampare tutti i campi del recorset
        While Not Rs1.EOF
                For contFields = 0 To (Rs1.Fields.Count-1)
                        Debug.Print Rs1.Fields(contFields).Name & " = " &
                        Rs1.Fields(contFields).Value
                Next
                Rs1.MoveNext
        Wend
        Rs1.Close
        Conn1.Close
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

GetColumn, DLRCmdTarget Function

Syntax GetColumn(_lpszName)

Description This function gets the column object specified by the lpszName parameter. The methods and the properties available for the returned object are described in the DLRColumnCmdTarget interface.

Parameter	Description
lpszName As String	Name of the column to be retrieved.

Result Object
If Function has been executed successfully it will retrieve an object of type DLRColumnCmdTarget if otherwise Nothing is returned.

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        If Not(objDLR.IsRecipe) Then
                Dim obj As DLRColumnCmdTarget
                Set obj = objDLR.GetColumn("Col00001")
                Debug.Print obj.Variable
        End If
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

GetColumnNameList, DLRCmdTarget Function

Syntax GetColumnNameList()

Description This function returns a string containing the list of columns defined for the Data Logger/Recipe. The filed columns are separated by commas and column names containing spaces will be shown between square brackets.

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRCmdTarget

Public Sub Click()
        Set objDLR = GetData LoggerRecipe("Recipe1")
        Debug.Print objDLR.GetColumnNameList 'return string like: [ID Ricetta],
        Column1, Column2, ...
        Set objDLR = Nothing
End Sub
```

GetDNSConnectionString, DLRCmdTarget Function

Syntax GetDNSConnectionString()

Description This function allows you to set or get a customized ODBC link. Movicon will creat a file in Access2000 format for default in the project's "LOGS" folder with the following name:

ProjectName_HisLog.mdb

By using this function you can customize the ODBC link by creating a different database with a different name than the one created for default by Movicon.

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        Debug.Print objDLR.GetDNSConnectionString
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

GetNextTickLocalTime, DLRCmdTarget Function

Syntax GetNextTickLocalTime()

Description This function returns the time and date in string format of the next recording on the Datalogger. The return value indicates the recording's date and time in Local time.

Parameter	Description
None	None

Result Date

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        Debug.Print objDLR.GetNextTickLocalTime
```

```
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

GetNextTickTime, DLRCmdTarget Function

Syntax GetNextTickTime()

Description This function returns the time and date of the next recording on the datalogger. The returned value indicates the data and time in GMT (Greenwich Mean Time)..

Parameter	Description
None	None

Result Date

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        Debug.Print objDLR.GetNextTickTime
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

ActivateVariable, DLRCmdTarget Property

Syntax ActivateVariable = _String

Description This property sets or returns the name of Movicon Real Time DB variable to be used for executing the activation of the selected recipe. By using this command the data loaded on the recipe's temporary variables will also be activated on the recipe's own real variables.
The variables will be set to the "zero" value by Movicon once the operation has been done.

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        If objDLR.IsRecipe Then
                Debug.Print objDLR.ActivateVariable
        End If
End Sub
```

```
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

CRWReportFile, DLRCmdTarget Property

Syntax CRWReportFile = _String

Description This property sets or returns the name of the CrystalReport file associated to the reference Datalogger/recipe.

This property is not supported in Windows CE. (If set, always returns an empty string)

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        Debug.Print objDLR.CRWReportFile
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

DeleteVariable, DLRCmdTarget Property

Syntax DeleteVariable = _String

Description This property sets or returns the name of the Movicon Real Time DB variable to be used for cancelling the selected recipe. The recipe will be deleted from the Database By using this command.
The variable will then be set with the "zero" value by Movicon once the operation has been done.

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        If objDLR.IsRecipe Then
                Debug.Print objDLR.DeleteVariable
        End If
```

```
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

DSN, DLRCmdTarget Property

Syntax DSN = _String

Description This setting permits you to set or retrieve a customized **ODBC** for the referenced Data Logger or Recipe. Movicon will created DSN for default using the same project name plus the "_DLR" suffix, configured for accessing the specified database using the project's "Default ODBC PlugIn". The name of the DSN will be:

ProjectName_DLR

This property can used to customize the **ODBC** connection, by creating a custom database that is different from the one defined in the project's "Default ODBC PlugIn".

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        Debug.Print objDLR.DSN
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

DurationDays, DLRCmdTarget Property

Syntax DurationDays = _Long

Description This property allows you to get or set how many days the data is to be stored before being recycled. To avoid creating tables overloaded with data, you must base the number of days entered on how frequent the data is recorded. For instance, more frequent the recordings, means the more the maximum recording time must be lowered.

The maximum recording time is inserted based on your own requirements, but you must take into account how frequent the recordings are, and the type of database that is going to be used. For example, if you are going to use a Access2000 database you will be more restricted with the quantity of recorded data it can hold in respect to a SQL Server database.

Parameter	Description
None	None

Result Long

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        Debug.Print objDLR.DurationDays
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

DurationHours, DLRCmdTarget Property

Syntax DurationHours = _Long

Description This property allows you to get or set the number hours, in addition to the days set in the DuarationDays property, the data is to be stored before being recycled. To avoid creating tables overloaded with data, you must base the number of days entered on how frequent the data is recorded. For instance, more frequent the recordings, means the more the maximum recording time must be lowered.

The maximum recording time is inserted based on your own requirements, but you must take into account how frequent the recordings are, and the type of database that is going to be used. For example, if you are going to use a Access2000 database you will be more restricted with the quantity of recorded data it can hold compared with a SQL Server database.

Parameter	Description
None	None

Result Long

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        Debug.Print objDLR.DurationHours
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

DurationMinutes, DLRCmdTarget Property

Syntax DurationMinutes = _Long

Description This property allows you to get or set the number of minutes, in addition to days set in the DurationDays and the hours set with the DurationHours property, the data is to be stored before being recycled. To avoid creating tables overloaded with data, you must base the number of days entered on how frequent the data is recorded. For instance, more frequent the recordings, means the more the maximum recording time must be lowered.

The maximum recording time is inserted based on your own requirements, but you must take into account how frequent the recordings are, and the type of database that is going to be used. For example, if you are going to use a Access2000 database you will be more restricted with the quantity of recorded data it can hold compared with a SQL Server database.

Parameter	Description
None	None

Result Long

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        Debug.Print objDLR.DurationMinutes
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

Enabled, DLRCmdTarget Property

Syntax Enabled = _Boolean

Description This property sets or returns the 'Enabled' property of the reference datalogger/recipe.

Parameter	Description
None	None

Result Boolean

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        Debug.Print objDLR.Enabled
End Sub
Public Sub SymbolLoading()
```

```
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

EnableTimeFrom, DLRCmdTarget Property

Syntax EnableTimeFrom = _Date

Description This property sets or returns the DataLogger's start time for recording data when the "Enable Day TimeFrame" option has been enabled.

Parameter	Description
None	None

Result Date

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        Debug.Print objDLR.EnableTimeFrom
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

EnableTimeFromTo, DLRCmdTarget Property

Syntax EnabledTimeFromTo = _Boolean

Description This property sets or returns the recording property within a day TimeFrame. When activating this property you will need to specify which day timeframe the data recording is allowed (this has effect only if the "Recording Variable" is true). Any recording commands given to the Data Logger object outside this timeframe will be ignored accept for those recordings carried out by the "Record on Command" variable which will remain active. This property is taken into consideration by Movicon only when the 'On Time' recording property has been enabled for the same datalogger; this property can be verified by means of the script code with the RecordOnTime property described in DLRCmdTarget interface.

The recording of data in TimeFrames, especially with those that change, can also be done by using the "Enable Recording Variable".

Parameter	Description
None	None

Result Boolean

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        Debug.Print objDLR.EnableTimeFromTo
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

EnableTimeTo, DLRCmdTarget Property

Syntax EnableTimeTo = _Date

Description This property sets or returns the Data Logger's recording data end time when the "Enable Day TimeFrame" is enabled.

Parameter	Description
None	None

Result Date

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
                Debug.Print objDLR.EnableTimeTo
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

EnableTimeVariable, DLRCmdTarget Property

Syntax EnableTimeVariable = _String

Description This property sets or returns the name of the Movicon Real Time DB variable to be used for executing the DataLogger to record on command. This selection is only valid when the "On Variable" has been enabled. The variable can be declared any type (bit, byte, word, etc.) since Movicon will execute the recording when the value contained in the variable is different from zero. As soon as the recording has taken place, Movicon will force the variable's value to zero.

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        Debug.Print objDLR.EnableTimeVariable
```

```
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

ExecuteVariable, DLRCmdTarget Property

Syntax ExecuteVariable = _String

Description This property sets or returns the name of the Movicon Real Time DB variable to be used for executing any query set for the recipe. The query to be executed must be contained inside the "Query Variable".
The variable will then be set to the "zero" value by Movicon once the operation has been done.

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        If objDLR.IsRecipe Then
                Debug.Print objDLR.ExecuteVariable
        End If
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

Filter, DLRCmdTarget Property

Syntax Filter = _String

Description This property gives you the possibility to use a recipe data filter (WHERE clause).

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        If objDLR.IsRecipe Then
                Debug.Print objDLR.Filter
        End If
```

```
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

FilterVariable, DLRCmdTarget Property

Syntax FilterVariable = _String

Description This property sets or returns the name of the Movicon Real Time DB variable to be used for exectuing the a recipe data filter (WHERE clause).

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        If objDLR.IsRecipe Then
                Debug.Print objDLR.FilterVariable
        End If
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

InsertVariable, DLRCmdTarget Property

Syntax InsertVariable = _String

Description This property sets or get the name of the Movicon Real Time DB variable to be used for saving the selected recipe. The data loaded into recipe's dummy variables will be saved on Database.
The variables will then be set to "zero" value by Movicon once the operation has been completed.

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        If objDLR.IsRecipe Then
                Debug.Print objDLR.InsertVariable
        End If
```

```
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

IsRecipe, DLRCmdTarget Property

Syntax IsRecipe = _Boolean

Description This property returns the True boolean when the reference DLRCmdTarget object type is a recipe. The value will return False when the object is a datalogger.

Parameter	Description
None	None

Result Boolean

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        Debug.Print objDLR.IsRecipe
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

LocalTimeColName, DLRCmdTarget Property

Syntax LocalTimeColName = _String

Description This property returns the name of the Data Logger table's Local Time Column. If a name is not specified, the default name will be used instead. The Local Time Column indicates the local date and time when recording took place.

This property is read only.

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        Debug.Print objDLR.LocalTimeColName
End Sub
Public Sub SymbolLoading()
```

```
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

MaxCacheBeforeFlush, DLRCmdTarget Property

Syntax MaxCacheBeforeFlush = _Long

Description This property sets or returns the Cache's maximum size before the system unloads the data on file. The number set in Bytes.

Parameter	Description
None	None

Result Long

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        Debug.Print objDLR.MaxCacheBeforeFlush
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

MaxError, DLRCmdTarget Property

Syntax MaxError = _Long

Description This property sets or returns the maximum number of DBMS errors that when exceeded the connection is considered not valid and the data is saved on file in ASCII format in the folders relating to the project ("DLOGGERS", "LOGS", "DATA").

Parameter	Description
None	None

Result Long

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        Debug.Print objDLR.MaxError
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

MaxNumberTrans, DLRCmdTarget Property

Syntax MaxNumberTrans = _Long

Description This property sets or returns the maximum number or transitions per cycle to be updated before they close.

Parameter	Description
None	None

Result Long

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        Debug.Print objDLR.MaxNumberTrans
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

MoveFirstVariable, DLRCmdTarget Property

Syntax MoveFirstVariable = _String

Description This property sets or returns the name of the Movicon Real Time DB variable to be used for moving the selected RecordSet data to the first record.
The rising edge of this variable allows the extracted values in the RecordSet to be represented in the variables associated to the Database Columns.
For instance, by filtering a group of data from the database by using the Filter command, a RecordSet will be created in memory containing the filtered data. When activating the "Move First Variable" with a value other than zero, the extracted value in the RecordSet relating to the first Record will be written in the variables associated to the database columns.
The variable will then be set with the "zero" value by Movicon once the operation has been completed.

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        If objDLR.IsRecipe Then
                Debug.Print objDLR.MoveFirstVariable
        End If
End Sub
```

```
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

MoveLastVariable, DLRCmdTarget Property

Syntax MoveLastVariable = _String

Description This property sets or returns the name of the Movicon Real Time DB variable to be used for moving the selected RecordSet data to the last record.
The rising edge of this variable allows the extracted values in the RecordSet to be represented in the variables associated to the Database Columns.
For instance, when filtering a group of data from the database by using the Filter command, a RecordSet will be created in memory containing the filtered data. When activating the "Move Last Variable" with a value different from zero, the extracted value in the RecordSet relating to the last Record will be written in the variables associated to the database columns.
The variable will then be set with the "zero" value by Movicon once the operation has been completed.

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        If objDLR.IsRecipe Then
                Debug.Print objDLR.MoveLastVariable
        End If
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

MoveNextVariable, DLRCmdTarget Property

Syntax MoveNextVariable = _String

Description This property sets or returns the name of the Movicon Real Time DB variable to be used for moving the selected RecordSet data to the next record in respect to the current one.
The rising edge of this variable allows the extracted values in the RecordSet to be represented in the variables associated to the Database Columns.
For instance, when filtering a group of data from the database by using the Filter command, a RecordSet will be created in memory containing the filtered data. When activating the "Move Next Variable" with a value different from zero, the extracted value in the RecordSet relating to the next Record, in respect to the current one, will be written in the variables associated to the database columns.
The variable will then be set with the "zero" value by Movicon once the operation has been completed.

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        If objDLR.IsRecipe Then
                Debug.Print objDLR.MoveNextVariable
        End If
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

MovePrevVariable, DLRCmdTarget Property

Syntax MovePrevVariable = _String

Description This property sets or returns the name of the Movicon Real Time DB variable to be used for moving the selected RecordSet data to the previous record in respect to the current one.
The rising edge of this variable allows the extracted values in the RecordSet to be represented in the variables associated to the Database Columns.
For instance, when filtering a group of data from the database by using the Filter command, a RecordSet will be created in memory containing the filtered data. When activating the "Move Previous Variable" on a value different from zero, the extracted value in the RecordSet relating to the previous Record, in respect to the current one, will be written in the variables associated to the database columns.
The variable will then be set with the "zero" value by Movicon once the operation has been executed.

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        If objDLR.IsRecipe Then
                Debug.Print objDLR.MovePrevVariable
        End If
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

MSecColName, DLRCmdTarget Property

Syntax MSecColName = _String

Description This property returns the name of the Data Logger table MSce Column. If no name has been specified, the default name will be used instead. The MSec Column indicates the milliseconds relating to time or recording.

This property is read only.

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        Debug.Print objDLR.MSecColName
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

Name, DLRCmdTarget Property

Syntax Name = _String

Description This property returns the name of the reference datalogger/recipe object.

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        Debug.Print objDLR.Name
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

PrintVariable, DLRCmdTarget Property

Syntax PrintVariable = _String

Description This property sets or returns the name of the Movicon Real Time DB variable that when set to a logic status different from zero will print the recorded data. The variable will then be set to the "zero value by Movicon once the operation has been done LThe print is however always referred to the Report specified in the Data Logger's "Report File" property. The report will be in Report Designer format (.repx) or Crystal Report (.RPT) and associated to the Data Logger's data table.
For further information on the Reports management and printing reports please consult the chapter on "Reports" of this document.

This property is not supported in Windows CE.(If set, always returns an empty string)

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        Debug.Print objDLR.PrintVariable
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

Query, DLRCmdTarget Property

Syntax Query = _String

Description This property sets or returns the query in standard SQL language on data to be deleted from the Database.

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        If objDLR.IsRecipe Then
                Debug.Print objDLR.Query
        End If
End Sub
Public Sub SymbolLoading()
```

```
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

QueryVariable, DLRCmdTarget Property

Syntax QueryVariable = _String

Description This property sets or returns the name of the Movicon Real Time DB variable to be used for executing a query in standard SQL language on data to be selected from the Database.

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        If objDLR.IsRecipe Then
                Debug.Print objDLR.QueryVariable
        End If
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

ReadVariable, DLRCmdTarget Property

Syntax ReadVariable= _String

Description This property sets or returns the name of the Movicon RealTimeDB variable to be used for executing the read command of reicpe values from the field and update the selected recipe's temporary variables with them.

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRCmdTarget

Public Sub Click()
        If objDLR.IsRecipe Then
                objDLR.ReadVariable= "VarNameUsedToRead"
        End If
End Sub
```

```
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("Recipe")
End Sub
```

ReasonColName, DLRCmdTarget Property

Syntax ReasonColName = _String

Description This property returns the name of Data Logger table Reason Column Name. If not specified, the default name will be used instead. The Reason Column indicates which event evoked the recording (command, change or time).

This property is read only.

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        Debug.Print objDLR.ReasonColName
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

RecipeIndexName, DLRCmdTarget Property

Syntax RecipeIndexName = _String

Description This property sets or returns the name of the recipe's index field.

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        If objDLR.IsRecipe Then
                Debug.Print objDLR.RecipeIndexName
        End If
End Sub
```

```
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

RecordOnChange, DLRCmdTarget Property

Syntax RecordOnChange = _Boolean

Description This property sets or returns the 'On Change" recording property of the reference datalogger. When the property returns the True boolean value this means that the datalogger records on status change of the variables associated to the columns. The record on status change does not influence the record on command or time, as all modalities can co-exist.
Movicon will record on each value change of the variables (columns) associated to the Data Logger, even when enabled to record on command or time. Therefore any changes to variables between each on command or on time recording will also be recorded.

The recording will be done in conformity with the recording enabling settings "Enable Recording Variable".

The precise functioning of the recording on change when the Data Logger's "Enable Recording Variable" has been inserted is as follows:

- Movicon records upon variable change of a variable associated to a column of the Data Logger when the Enabling variable is different from zero
- Movicon records on the rising edge of the enabling variable, when a variable associated to a Data Logger column changes to a different value from the last recording carried out by Movicon
- Movicon DOES NOT record anything when the enabling flag is at zero value
- Movicon DOES NOT record on the rising edge of the enabling flag when all the variables associated to the Data Logger columns have not changed since the last recording was made. This applies even when the variables change during Data Logger disablement, but then return to the values last recorded

Parameter	Description
None	None

Result Boolean

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        Debug.Print objDLR.RecordOnVariable
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

RecordOnChangeDeadBand, DLRCmdTarget Property

Syntax RecordOnChangeDeadBand = _Double

Description This property sets or returns the dead band value for the DataLogger desired.

Parameter	Description
None	None

Result Double

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        Debug.Print CStr(objDLR.RecordOnChangeDeadBand)
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

RecordOnChangeDeadBandPercent, DLRCmdTarget Property

Syntax RecordOnChangeDeadBandPercent = _Boolean

Description This property enables or disables the dead band control in percentages for the DataLogger desired.

Parameter	Description
None	None

Result Boolean

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        Debug.Print CStr(objDLR.RecordOnChangeDeadBandPercent)
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

RecordOnChangeEnableDeadBand, DLRCmdTarget Property

Syntax RecordOnChangeEnableDeadBand = _Boolean

Description This property enabled or disables the dead band management for the DataLogger desired.

Parameter	Description
None	None

Result Boolean

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        Debug.Print CStr(objDLR.RecordOnChangeEnableDeadBand)
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

RecordOnlyWhenQualityGood, DLRCmdTarget Property

Syntax RecordOnlyWhenQualityGood = _Boolean

Description This property allows you to enable the recording of data to take place only when all the variables associated to the Data Logger have Good 'Quality' properties. When this setting is left at False the recording will take place independently from the quality of the variables.

Parameter	Description
None	None

Result Boolean

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        Debug.Print objDLR.RecordOnlyWhenQualityGood
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

RecordOnTime, DLRCmdTarget Property

Syntax RecordOnTime = _Boolean

Description This property sets or returns the 'on time" recording property of the reference datalogger.

Parameter	Description
None	None

Result Boolean

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        Debug.Print objDLR.RecordOnTime
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

RecordOnVariable, DLRCmdTarget Property

Syntax RecordOnVariable = _Boolean

Description This property sets or returns the 'on command' recording property of the reference datalogger. When the property returns the True boolean value this means that the datalogger only records when the associated variable passes from the "zero" to a value "higher than zero". The variable in question must belong to the Movicon Real Time DB and can be declared as any type (bit, byte, word, etc) as Movicon executes the recording when the value contained in the variable is different from zero. Movicon will force the value of the recording variable to zero after the recording has taken place.

Parameter	Description
None	None

Result Boolean

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        Debug.Print objDLR.RecordOnVariable
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

RecVariable, DLRCmdTarget Property

Syntax RecVariable = _String

Description This property sets or returns the name of the associated variable for recording on command.

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        Debug.Print objDLR.RecVariable
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

RecycleDBConnection, DLRCmdTarget Property

Syntax RecycleDBConnection = _Boolean

Description This property, when enabled, allows the **DBMS** connection to be kept open and used for all future transitions. When this property is disabled, the **DBMS** connection will open when a transition is requested and will close again afterwards.

It would be handy to disable the "Keep the DB Connection open" only in cases where the recordings are less frequent.

Parameter	Description
None	None

Result Boolean

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        Debug.Print objDLR.RecycleDBConnection
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

ResetVariable, DLRCmdTarget Property

Syntax ResetVariable = _String

Description This property sets or returns the name of the Movicon Real Time DB variable which, when set at a logic status 'different from zero', will cancel all the values in the table recorded up till that moment inorder to start a new recordset. The variable will then be reset to 'zero' value by Movicon once this operation has been executed.

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        Debug.Print objDLR.ResetVariable
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

Sort, DLRCmdTarget Property

Syntax Sort = _String

Description This property sets or returns the name of a Movicon Real Time DB variable to be used for sorting recipe data (ORDER BY clause).

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        If objDLR.IsRecipe Then
                Debug.Print objDLR.Sort
        End If
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

SortVariable, DLRCmdTarget Property

Syntax SortVariable = _String

Description This property sets or returns the Movicon Real Time DB variable to be used for sorting recipe data into order (ORDER BY clause).

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        If objDLR.IsRecipe Then
                Debug.Print objDLR.SortVariable
        End If
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

StatusVariable, DLRCmdTarget Property

Syntax StatusVariable = _String

Description This property sets or returns the name of a Movicon Realtime DB variable which has the returned execution status of any query that may have been carried out. The moment in which a query is executed, Movicon notifies the logic of the query's execution status by using the following bits of the status variable:

Bit 0 = query in execution
Bit 1 = BOF (Beginning Of File)
Bit 2 = EOF (End of File)
Bit 3 = Deleted Record
Bit 4 = Error

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        If objDLR.IsRecipe Then
                Debug.Print objDLR.StatusVariable
        End If
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
```

```
End Sub
```

TableName, DLRCmdTarget Property

Syntax TableName = _String

Description This property returns the name of the table associated to the reference Datalogger/Recipe.

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        Debug.Print objDLR.TableName
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

TimeColName, DLRCmdTarget Property

Syntax TimeColName = _String

Description This property returns the name of Data Logger Table's Time Column. When this is left blank the default name will be used instead. The Time Column indicated the data and time of the recording in GMT (Greenwich Mean Time).

This property is read only.

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        Debug.Print objDLR.TimeColName
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

TimeRecHour, DLRCmdTarget Property

Syntax TimeRecHour = _Byte

Description This property sets or returns the DataLogger's Sampling time in hours. This value has meaning only when the record "On Time" is enabled.

Parameter	Description
None	None

Result Byte

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        Debug.Print objDLR.TimeRecHour
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

TimeRecMin, DLRCmdTarget Property

Syntax TimeRecMin = _Byte

Description This property sets or returns the DataLogger's Sampling time in minutes. This value has meaning only when the record "On Time" is enabled.

Parameter	Description
None	None

Result Byte

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        Debug.Print objDLR.TimeRecMin
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

TimeRecMSec, DLRCmdTarget Property

Syntax TimeRecMSec = _Integer

Description This property sets or returns the DataLogger's Sampling time in milliseconds. This value has meaning only when the record "On Time" is enabled.

Parameter	Description
None	None

Result Integer

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        Debug.Print objDLR.TimeRecMSec
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

TimeRecSec, DLRCmdTarget Property

Syntax TimeRecSec = _Byte

Description This property sets or returns the DataLogger's Sampling time in seconds. This value has meaning only when the record "On Time" is enabled.

Parameter	Description
None	None

Result Byte

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        Debug.Print objDLR.TimeRecSec
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

UseIMDB,DLRCmdTarget property

Syntax UseIMDB = _Boolean

Description This property is read only and lets you know if the data logger is set fro recording values using the InMemoryDataBase engine.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Main
Dim obj As DLRCmdTarget

Set obj = GetDataLoggerRecipe("DataLogger")
If Not obj Is Nothing Then
MsgBox "UseIMDB = " & obj.UseIMDB
End If
End Sub
```

UserColName, DLRCmdTarget Property

Syntax UserColName = _String

Description Ths property returns the name of the Data Logger tables User Column. If no name has been specified the default name will be used instead. The User Column indicated the name of the user who was active at time or recording.

This property is read only.

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        Debug.Print objDLR.UserColName
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

UserName, DLRCmdTarget Property

Syntax UserName = _String

Description This property sets or returns the user name used for the **ODBC** connection.

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        Debug.Print objDLR.UserName
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

VarCharsMax, DLRCmdTarget Property

Syntax VarCharsMax = _Long

Description This property sets or returns the maximum number of characters for the string type column. The number set represents the string's number of characters.

Parameter	Description
None	None

Result Long

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        Debug.Print objDLR.VarCharsMax
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

27. DLRColumnCmdTarget

AddNumUpdatesCol, DLRColumnCmdTarget Property

Syntax AddNumUpdatesCol = _Boolean

Description This property enables or disables the 'Add Num.Updates Column' property; when enabled this allows you to add a column to the Database table which refers to the variable returning the number of changes the variable underwent in the interval time between one recording and the next.

Parameter	Description
None	None

Result Boolean

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        If Not(objDLR.IsRecipe) Then
                Dim obj As DLRColumnCmdTarget
                Set obj = objDLR.GetColumn("Col00001")
                Debug.Print obj.AddNumUpdatesCol
        End If
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

AddQualityColumn, DLRColumnCmdTarget Property

Syntax AddQualityColumn = _Boolean

Description This property enables or disables the 'Add Quality Column' property; when enabled a column will be added to the Database table referring to the variable which returns its Quality status when the recording goes into execution.

Parameter	Description
None	None

Result Boolean

Example:

```
Dim objDLR As DLRCmdTarget
   Public Sub Click()
            If Not(objDLR.IsRecipe) Then
                     Dim obj As DLRColumnCmdTarget
                     Set obj = objDLR.GetColumn("Col00001")
                     Debug.Print obj.AddQualityColumn
            End If
   End Sub
   Public Sub SymbolLoading()
            Set objDLR = GetDataLoggerRecipe("DLR1")
   End Sub
```

Name, DLRColumnCmdTarget Property

Syntax Name = _String

Description This property sets or returns the name of the column associated to the reference datalogger/recipe.

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRCmdTarget
   Public Sub Click()
            Dim obj As DLRColumnCmdTarget
            Set obj = objDLR.GetColumn("Col00001")
            Debug.Print obj.Name
   End Sub
   Public Sub SymbolLoading()
            Set objDLR = GetDataLoggerRecipe("DLR1")
   End Sub
```

NumUpdatesColumnName, DLRColumnCmdTarget Property

Syntax NumUpdatesColumnName = _String

Description This property allows the Num.Updates Column to be set with a customized name when the "Add Num.Updates Column' property has been enabled or when the AddNumUpdatesCol has been set to True. When this field is left blank the default name (ColumnName_NumUpdates) will be used instead.

Parameter	Description

None	None

Result String

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        If Not(objDLR.IsRecipe) Then
                Dim obj As DLRColumnCmdTarget
                Set obj = objDLR.GetColumn("Col00001")
                Debug.Print obj.NumUpdatesColumnName
        End If
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

QualityColumnName, DLRColumnCmdTarget Property

Syntax QualityColumnName = _String

Description This property allows the Quality column to be set with a customised name when the 'Add Quality Column' property has been enabled or when the AddQualityColumn property has been set to True. When this field is left empty the default name (ColuimnName_Quality) will be used instead.

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        If Not(objDLR.IsRecipe) Then
                Dim obj As DLRColumnCmdTarget
                Set obj = objDLR.GetColumn("Col00001")
                Debug.Print obj.QualityColumnName
        End If
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

RecipeIndex, DLRColumnCmdTarget Property

Syntax RecipeIndex = _Boolean

Description This property is used for defining whether the column in question is to be the recipe index, meaning the column which identifies the recipe's contents or ingredients. Each recipe can be set with one recipe index only.

Parameter	Description
None	None

Result Boolean

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        If (objDLR.IsRecipe) Then
                Dim obj As DLRColumnCmdTarget
                Set obj = objDLR.GetColumn("Col00001")
                Debug.Print obj.RecipeIndex
        End If
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

RecipeTempVariable, DLRColumnCmdTarget Property

Syntax RecipeTempVariable = _String

Description This property sets or returns the name of the Movicon Real Time DB variable which is to be used as the Temporary variable for keeping recipe data from the DB. Only when the recipe's activation command is executed will the value contained in the "Temporary Variable" be copied into the recipe's variable.

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        If (objDLR.IsRecipe) Then
                Dim obj As DLRColumnCmdTarget
                Set obj = objDLR.GetColumn("Col00001")
                Debug.Print obj.RecipeTempVariable
        End If
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

RecordType, DLRColumnCmdTarget Property

Syntax RecordType = _String

Description This property allows you to set which variable value is to be recorded on database.
The possibilities are as follows:

- **Instantaneous:** the variable's instantaneous value will be recorded, which is the value contained in the variable the instant the recording take place.
- **Minimum:** the minimum value, obtained by the variable in the interval time between on recording and the next, will be recorded.
- **Maximum:** the maximum value, obtained by the variable in the interval time between on recording and the next, will be recorded.
- **Average:** the average value, obtained by the variable in the interval time between on recording and the next, will be recorded.

The values are:

Instantaneous - 0
Minimum - 1
Maximum - 2
Average - 3

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        If Not(objDLR.IsRecipe) Then
                Dim obj As DLRColumnCmdTarget
                Set obj = objDLR.GetColumn("Col00001")
                Debug.Print obj.RecordType
        End If
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

StatisticAverageValue, DLRColumnCmdTarget Property

Syntax StatisticAverageValue = _Variant

Description This property returns the statistic average value associated to the column.

Parameter	Description
None	None

Result Variant

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        If Not(objDLR.IsRecipe) Then
                Dim obj As DLRColumnCmdTarget
                Set obj = objDLR.GetColumn("Col00001")
                Debug.Print obj.StatisticAverageValue
        End If
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

StatisticMaxValue, DLRColumnCmdTarget Property

Syntax StatisticMaxValue = _Variant

Description This property returns the statistic max value of the variable associated to the column.

Parameter	Description
None	None

Result Variant

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        If Not(objDLR.IsRecipe) Then
                Dim obj As DLRColumnCmdTarget
                Set obj = objDLR.GetColumn("Col00001")
                Debug.Print obj.StatisticMaxValue
        End If
End Sub
Public Sub SymbolLoading()
```

```
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

StatisticMinValue, DLRColumnCmdTarget Property

Syntax StatisticMinValue = _Variant

Description This property returns the statistic minimum value of the variable associated to the column.

Parameter	Description
None	None

Result Variant

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        If Not(objDLR.IsRecipe) Then
                Dim obj As DLRColumnCmdTarget
                Set obj = objDLR.GetColumn("Col00001")
                Debug.Print obj.StatisticMinValue
        End If
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

StatisticNumUpdates, DLRColumnCmdTarget Property

Syntax StatisticNumUpdates = _Variant

Description This property returns the number of udates of the variable associated to the column.

Parameter	Description
None	None

Result Variant

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        If Not(objDLR.IsRecipe) Then
                Dim obj As DLRColumnCmdTarget
```

```
                Set obj = objDLR.GetColumn("Col00001")
                Debug.Print obj.StatisticNumUpdates
        End If
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

Variable, DLRColumnCmdTarget Property

Syntax Variable = _String

Description This property,in ready only, sets or returns the name of the variable associated to the column of the reference datalogger/recipe.

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRCmdTarget
Public Sub Click()
        Dim obj As DLRColumnCmdTarget
        Set obj = objDLR.GetColumn("Col00001")
        Debug.Print obj.Variable ' i.e.: return string "VAR00001"
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetDataLoggerRecipe("DLR1")
End Sub
```

28. DLRWndCmdTarget

OnFilter, DLRWndCmdTarget Event

Description Event occurs each time a request is made to apply a filter for extracting data from the datalogger.

Parameter	Description
bRet As Boolean	Enabling upon status change.

OnPrint, DLRWndCmdTarget Event

Description Event occurs each time a request is made to print data loaded in the displayed window.

This event is not supported in Windows CE.

Parameter	Description
bRet As Boolean	Enabling upon status change.

OnRefresh, DLRWndCmdTarget Event

Description Event occurs each time a request is made to refresh the data loaded in the display window.

Parameter	Description
bRet As Boolean	Enabling upon status change.

EditCopy, DLRWndCmdTarget Function

Syntax EditCopy()

Description This property executes a copy of the selected line contents to the clipboard.

Parameter	Description

None	None

Result Boolean

Example:

```
Dim objDLR As DLRWndCmdTarget
Public Sub Click()
        Debug.Print objDLR.EditCopy
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetSynopticObject.GetSubObject("DLRWindow").GetObjectInterface
End Sub
```

EditLayout, DLRWndCmdTarget Function

Syntax EditLayout()

Description This function opens the configuration window of the fields to be displayed in the DataLogger Window.

This function is only executed if the "Show Control window" property has been enabled in the Window object. Otherwise the "Field Choice Window" will not open and this function will return the "False" value.

Parameter	Description
None	None

Result Boolean

Example:

```
Dim objDLR As DLRWndCmdTarget
Public Sub Click()
        Debug.Print objDLR.EditLayout
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetSynopticObject.GetSubObject("DLRWindow").GetObjectInterface
End Sub
```

LoadExtSettings, DLRWndCmdTarget Function

Syntax LoadExtSettings

Description This function permits the object's relating external file settings to be loaded. This file can be specified in design mode in the "External File settings" property or in the "ExtSettingsFile" interface properties. The extension provided for this file is ".SXML".

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
Dim objSymbol As DLRWndCmdTarget
Set objSymbol = GetSynopticObject.GetSubObject("TestObject").GetObjectInterface
If objSymbol Is Nothing Then Exit Sub
objSymbol.ExtSettingsFile = "test.sxml"
objSymbol.LoadExtSettings
Set objSymbol = Nothing
End Sub
```

RecalcLayout, DLRWndCmdTarget Function

Syntax RecalcLayout()

Description This function executes a recalculation on the object's layout. This function needs to be executed after a property has been changed involving the object's layout where, for example, a column has been added or taken away.

Parameter	Description
None	None

Result Boolean

Example:

```
Dim objDLR As DLRWndCmdTarget
Public Sub Click()
        objDLR.RecalcLayout
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetSynopticObject.GetSubObject("DLRWindow").GetObjectInterface
End Sub
```

Refresh, DLRWndCmdTarget Function

Syntax Refresh()

Description This function refreshes data being displayed in the object. This function needs to be carried out after the interrogation query on the database has been changed.

Parameter	Description
None	None

Result Boolean

Example:

```
Dim objDLR As DLRWndCmdTarget
Public Sub Click()
        Debug.Print objDLR.Refresh
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetSynopticObject.GetSubObject("DLRWindow").GetObjectInterface
End Sub
```

SaveExtSettings, DLRWndCmdTarget Function

Syntax SaveExtSettings

Description This function permits the objects settings to be save in the relating external settings file. This file can be specified when in design mode in the "Ext. Settings File" property, or using the property from the "ExtSettingsFile" interface. The extension provided for this file is ".SXML".

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
Dim objSymbol As DLRWndCmdTarget
Set objSymbol = GetSynopticObject.GetSubObject("TestObject").GetObjectInterface
If objSymbol Is Nothing Then Exit Sub
objSymbol.ExtSettingsFile = "test.sxml"
objSymbol.SaveExtSettings
Set objSymbol = Nothing
End Sub
```

AutoLayout, DLRWndCmdTarget Property

Syntax AutoLayout = _Boolean

Description When this property is enabled, the list layout will be set in automatic mode. This means that the table columns will be automatically resized so that all of them can be seen within the DataLogger Window. When this property is disabled, the window will open showing the columns with the sizes they were set with in the programming phase where the last ones might not be visible unless the horizontal scroll bar is used.

Parameter	Description
None	None

Result Boolean

Example:

```
Dim objDLR As DLRWndCmdTarget
Public Sub Click()
	Debug.Print objDLR.AutoLayout
End Sub
Public Sub SymbolLoading()
	Set objDLR = GetSynopticObject.GetSubObject("DLRWindow").GetObjectInterface
End Sub
```

ButtonPos, DLRWndCmdTarget Property

Syntax ButtonPos = _Integer

Description This setting returns the position where the data display window's buttons are to appear.

The options are:
0 = left
1 = top
2 = right
3 = bottom

Parameter	Description
None	None

Result Integer

Example:

```
Dim objDLR As DLRWndCmdTarget
Public Sub Click()
	If Not objDLR Is Nothing Then
		MsgBox "objDLR 's ButtonPos is " & objDLR.ButtonPos,vbInformation,GetProjectTitle
		objDLR.ButtonPos = 2
		objDLR.RecalcLayout
	Else
		MsgBox "objDLRWnd is nothing",vbInformation,GetProjectTitle
	End If
End Sub
Public Sub SymbolLoading()
	Set objDLR = GetSynopticObject.GetSubObject("DLRWindow").GetObjectInterface
End Sub
```

ButtonSize, DLRWndCmdTarget Property

Syntax ButtonSize = _Integer

Description This setting returns the size of the buttons which are to be displayed in the DataLogger Window.

The options are:
0 = small
1 = medium
2 = large

Parameter	Description
None	None

Result Integer

Example:

```
Dim objDLR As DLRWndCmdTarget
Public Sub Click()
	If Not objDLR Is Nothing Then
		MsgBox "objDLR 's ButtonSize is " & objDLR.ButtonSize,vbInformation,GetProjectTitle
		objDLR.ButtonSize = 2
		objDLR.RecalcLayout
	Else
		MsgBox "objDLRWnd is nothing",vbInformation,GetProjectTitle
	End If
End Sub
Public Sub SymbolLoading()
	Set objDLR = GetSynopticObject.GetSubObject("DLRWindow").GetObjectInterface
End Sub
```

Clickable, DLRWndCmdTarget Property

Syntax Clickable = _Boolean

Description This property lets you define whether the operator can interact with the DataLogger Window or not. When this property is set at False it will not be possible to use the mouse or the keyboard to control or manage any anything in this window such as sorting columns in order, viewing any help and using any of the commands that may be in the window.

Parameter	Description
None	None

Result Boolean

Example:

```
Dim objDLR As DLRWndCmdTarget
Public Sub Click()
	Debug.Print objDLR.Project
End Sub
Public Sub SymbolLoading()
	Set objDLR = GetSynopticObject.GetSubObject("DLRWindow").GetObjectInterface
End Sub
```

DLR, DLRWndCmdTarget Property

Syntax DLR = _String

Description This property sets or returns the name of the datalogger associated to the DataLogger Window.

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRWndCmdTarget
Public Sub Click()
        Debug.Print objDLR.DLR
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetSynopticObject.GetSubObject("DLRWindow").GetObjectInterface
End Sub
```

ExtSettingsFile, DLRWndCmdTarget Property

Syntax ExtSettingsFile = _String

Description This property sets or returns the external configuration file for the referenced object. the file can be alsow specified in design mode in the object's "Configuration File" property. The extension provided for this file is ".SXML".

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
Dim objSymbol As DLRWndCmdTarget
Set objSymbol = GetSynopticObject.GetSubObject("TestObject").GetObjectInterface
If objSymbol Is Nothing Then Exit Sub
objSymbol.ExtSettingsFile = "test.sxml"
objSymbol.SaveExtSettings
Set objSymbol= Nothing
End Sub
```

FilterBtnText, DLRWndCmdTarget Property

Syntax FilterBtnText = _String

Description This property sets or returns a text for the command button used for printing the data displayed in the data logger window. If nothing is specified in this property, Movicon will use the default text.

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRWndCmdTarget
Public Sub Click()
	If Not objDLR Is Nothing Then
		MsgBox "objDLR 's FilterBtnText is " & objDLR.FilterBtnText
		,vbInformation,GetProjectTitle
		objDLR.FilterBtnText = "Filter data"
		objDLR.Refresh
	Else
		MsgBox "objDLRWnd is nothing",vbInformation,GetProjectTitle
	End If
End Sub
Public Sub SymbolLoading()
	Set objDLR = GetSynopticObject.GetSubObject("DLRWindow").GetObjectInterface
End Sub
```

FilterFromDate, DLRWndCmdTarget Property

Syntax FilterFromDate = _Date

Description This property sets or returns the 'From Date' filter for displaying messages in the Movicon Trace window.

Parameter	Description
None	None

Result Date

Example:

```
Dim objDLR As DLRWndCmdTarget
Public Sub Click()
	If Not objDLR Is Nothing Then
		MsgBox "objDLR 's FilterFromDate is " &
		objDLR.FilterFromDate,vbInformation,GetProjectTitle
		objDLR.FilterFromDate = Now
		objDLR.Refresh
	Else
		MsgBox "objDLRWnd is nothing",vbInformation,GetProjectTitle
	End If
End Sub
Public Sub SymbolLoading()
	Set objDLR = GetSynopticObject.GetSubObject("DLRWindow").GetObjectInterface
End Sub
```

FilterToDate, DLRWndCmdTarget Property

Syntax FilterToDate = _Date

Description This property sets or returns the 'Data finale' filter for displaying messages in the Movicon Trace window.

Parameter	Description
None	None

Result Date

Example:

```
Dim objDLR As DLRWndCmdTarget
Public Sub Click()
	If Not objDLR Is Nothing Then
		MsgBox "objDLR 's FilterToDate is " & objDLR.FilterToDate
		,vbInformation,GetProjectTitle
		objDLR.FilterToDate = Now
		objDLR.Refresh
	Else
		MsgBox "objDLRWnd is nothing",vbInformation,GetProjectTitle
	End If
End Sub
Public Sub SymbolLoading()
	Set objDLR = GetSynopticObject.GetSubObject("DLRWindow").GetObjectInterface
End Sub
```

FilterUser, DLRWndCmdTarget Property

Syntax FilterUser = _String

Description This property sets or returns the 'Utente' filter for displaying messages in the Movicon Trace window.

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRWndCmdTarget
Public Sub Click()
	If Not objDLR Is Nothing Then
		MsgBox "objDLR 's FilterUser is " &
		objDLR.FilterUser,vbInformation,GetProjectTitle
		objDLR.FilterUser = "User00001"
		objDLR.Refresh
```

```
        Else
                MsgBox "objDLRWnd is nothing",vbInformation,GetProjectTitle
        End If
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetSynopticObject.GetSubObject("DLRWindow").GetObjectInterface
End Sub
```

GraphicButtons, DLRWndCmdTarget Property

Syntax GraphicButtons = _Boolean

Description When Enabling this property, the DataLogger-Recipe Window buttons are drawn using an icon instead of text. The text will instead be displayed as a tooltip when positioning the mouse on top of the button.

The tooltip is not managed in Windows CE versions.

This property is not managed by the 'Alarm Banner' object.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Click()
        GraphicButtons = True
        RecalcLayout
End Sub
```

IncludeMilliseconds, DLRWndCmdTarget Property

Syntax IncludeMilliseconds = _Boolean

Description By setting this property to True, the milliseconds will also be included in the 'Time' format of the window's columns supporting this data type.

Parameter	Description
None	None

Result Boolean

Example:

```
Dim objDLR As DLRWndCmdTarget
Public Sub Click()
        Debug.Print objDLR.IncludeMilliseconds
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetSynopticObject.GetSubObject("DLRWindow").GetObjectInterface
End Sub
```

MaxCount, DLRWndCmdTarget Property

Syntax MaxCount = _Long

Description This property sets or returns the maximum number of Rows that the DataLogger Window can display.

Parameter	Description
None	None

Result Long

Example:

```
Dim objDLR As DLRWndCmdTarget
Public Sub Click()
        Debug.Print objDLR.MaxCount
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetSynopticObject.GetSubObject("DLRWindow").GetObjectInterface
End Sub
```

NetworkBackupServerName, DLRWndCmdTarget Property

Syntax NetworkBackupServerName = _String

Description This property sets or returns the name of any Network Backup Server used for retrieving data to be displayed in the DataLogger/Recipe window when the Primary Server, being the one set in the "NetowrkServerName" property,is in timeout.

To display data from a Server you need to have the Data Logger/Recipe on the Client as well so that the Database structure can be retrieved. However, the Data Logger/Recipe can be created only as a structure and therefore does not need any variables associated to the columns.

Parameter	Description
None	None

Result String

Example:

```
Dim objDLRWnd As DLRWndCmdTarget
Public Sub Click()
        Debug.Print objDLRWnd.NetworkBackupServerName
End Sub
Public Sub SymbolLoading()
        Set objDLRWnd =
        GetSynopticObject.GetSubObject("DLRWindow").GetObjectInterface
End Sub
```

NetworkServerName, DLRWndCmdTarget Property

Syntax NetworkServerName = _String

Description This property returns the name of any Network Server where data is to be retrieved for displaying in the DataLogger/Recipe.

To display data from a Server you need to have the Data Logger/Recipe on the Client as well so that the Database structure can be retrieved. However, the Data Logger/Recipe can be created only as a structure and therefore does not need any variables associated to the columns.

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRWndCmdTarget
Public Sub Click()
        Debug.Print objDLR.NetworkServerName
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetSynopticObject.GetSubObject("DLRWindow").GetObjectInterface
End Sub
```

PrintBtnText, DLRWndCmdTarget Property

Syntax PrintBtnText = _String

Description This property sets or returns a text for the Print command button to print data in the data logger display window. When nothing is specified, Movicon will use the defualt text.

This property is not supported in Windows CE.(If set, always returns an empty string)

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRWndCmdTarget
Public Sub Click()
	If Not objDLR Is Nothing Then
		MsgBox "objDLR 's PrintBtnText is " & objDLR.PrintBtnText
		,vbInformation,GetProjectTitle
		objDLR.PrintBtnText = "Print data"
		objDLR.Refresh
	Else
		MsgBox "objDLRWnd is nothing",vbInformation,GetProjectTitle
	End If
End Sub
Public Sub SymbolLoading()
	Set objDLR = GetSynopticObject.GetSubObject("DLRWindow").GetObjectInterface
End Sub
```

Project, DLRWndCmdTarget Property

Syntax Project = _String

Description This property sets or returns the name of the child project from which you want to retrieve the data to be displayed. When this field is left blank, the current project will be used instead.

Only the name of the child project of the current project can be entered in this property.

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRWndCmdTarget
Public Sub Click()
	Debug.Print objDLR.Project
End Sub
Public Sub SymbolLoading()
	Set objDLR = GetSynopticObject.GetSubObject("DLRWindow").GetObjectInterface
End Sub
```

Query, DLRWndCmdTarget Property

Syntax Query = _String

Description This property is used for setting a query in customized standard SQL language for extracting requesting data in the database.

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRWndCmdTarget
Public Sub Click()
        objDLR.Query = "SELECT * FROM TabellaDLR WHERE ColVAR0001 >= 5"
        objDLR.Refresh
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetSynopticObject.GetSubObject("DLRWindow").GetObjectInterface
End Sub
```

RefreshBtnText, DLRWndCmdTarget Property

Syntax RefreshBtnText = _String

Description This property sets or returns a text for the command button which refreshes data displayed in the data logger window. When nothing is specified, Movicon will use the default text.

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRWndCmdTarget
Public Sub Click()
        If Not objDLR Is Nothing Then
                MsgBox "objDLR 's RefreshBtnText is " &
                objDLR.RefreshBtnText,vbInformation,GetProjectTitle
                objDLR.RefreshBtnText = "Refresh data"
                objDLR.Refresh
        Else
                MsgBox "objDLRWnd is nothing",vbInformation,GetProjectTitle
        End If
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetSynopticObject.GetSubObject("DLRWindow").GetObjectInterface
End Sub
```

ShowFilterBtn, DLRWndCmdTarget Property

Syntax ShowFilterBtn = _Boolean

Description This property allows you to displaly the command button for filtering data in the Data Logger window.

Parameter	Description
None	None

Result Boolean

Example:

```
Dim objDLR As DLRWndCmdTarget
Public Sub Click()
        If Not objDLR Is Nothing Then
                MsgBox "objDLR 's ShowFilterBtn is " & objDLR.ShowFilterBtn
                ,vbInformation,GetProjectTitle
                objDLR.ShowFilterBtn = Not objDLR.ShowFilterBtn
                objDLR.RecalcLayout
        Else
                MsgBox "objDLRWnd is nothing",vbInformation,GetProjectTitle
        End If
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetSynopticObject.GetSubObject("DLRWindow").GetObjectInterface
End Sub
```

ShowPrintBtn, DLRWndCmdTarget Property

Syntax ShowPrintBtn = _Boolean

Description This property allows the command button to be shown for printing data from the Data Logger window.
Print is done using the report file which should be specified in the "Report File" property. Movicon passes the same filter settings to the report for printing data, which coincide with that displayed in the window in question.

This property is not supported in Windows CE.(If set,always returns 'false')

Parameter	Description
None	None

Result Boolean

Example:

```
Dim objDLR As DLRWndCmdTarget
Public Sub Click()
        If Not objDLR Is Nothing Then
```

```
                MsgBox "objDLR 's ShowPrintBtn is " & objDLR.ShowPrintBtn
                ,vbInformation,GetProjectTitle
                objDLR.ShowPrintBtn = Not objDLR.ShowPrintBtn
                objDLR.RecalcLayout
        Else
                MsgBox "objDLRWnd is nothing",vbInformation,GetProjectTitle
        End If
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetSynopticObject.GetSubObject("DLRWindow").GetObjectInterface
End Sub
```

ShowRefreshBtn, DLRWndCmdTarget Property

Syntax ShowRefreshBtn = _Boolean

Description This property allows you to show the command button for refreshing data in the Data Logger display window.

Parameter	Description
None	None

Result Boolean

Example:

```
Dim objDLR As DLRWndCmdTarget
Public Sub Click()
        If Not objDLR Is Nothing Then
                MsgBox "objDLR 's ShowRefreshBtn is " & objDLR.ShowRefreshBtn
                ,vbInformation,GetProjectTitle
                objDLR.ShowRefreshBtn = Not objDLR.ShowRefreshBtn
                objDLR.RecalcLayout
        Else
                MsgBox "objDLRWnd is nothing",vbInformation,GetProjectTitle
        End If
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetSynopticObject.GetSubObject("DLRWindow").GetObjectInterface
End Sub
```

SortBy, DLRWndCmdTarget Property

Syntax SortBy = _String

Description This property sets or returns the 'Ordina Per' filter for displaying messages in the Movicon Data Logger window.

The possible fields are:

Col
LocalCol
MSecCol

ReasonCol
TimeCol
UserCol

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRWndCmdTarget
Public Sub Click()
	If Not objDLR Is Nothing Then
		MsgBox "objDLR 's SortBy is " & objDLR.SortBy ,vbInformation,GetProjectTitle
		objDLR.SortBy = "Col"
		objDLR.Refresh
	Else
		MsgBox "objDLRWnd is nothing",vbInformation,GetProjectTitle
	End If
End Sub
Public Sub SymbolLoading()
	Set objDLR = GetSynopticObject.GetSubObject("DLRWindow").GetObjectInterface
End Sub
```

SubItemReason, DLRWndCmdTarget Property

Syntax SubItemReason = _String

Description This property allows you to set the text to appear as the name for the "Reason" Column. When this field is left blank the default text will be used instead.

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRWndCmdTarget
Public Sub Click()
	Debug.Print objDLR.SubItemReason
End Sub
Public Sub SymbolLoading()
	Set objDLR = GetSynopticObject.GetSubObject("DLRWindow").GetObjectInterface
End Sub
```

SubItemReasonPos, DLRWndCmdTarget Property

Syntax SubItemReasonPos = _Integer

Description This property sets or returns the position of the "Reason" column within the Trace DB window. When setting a new value, the other columns will be automatically re-positioned in the window layout. In addition when setting the "-1", the column will be hidden. The "0" value is used to indicate position of the first column on the left in the window.

Parameter	Description
None	None

Result Integer

Example:

```
Dim objDLR As DLRWndCmdTarget
Public Sub Click()
        Debug.Print objDLR.SubItemReasonPos
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetSynopticObject.GetSubObject("DLRWindow").GetObjectInterface
End Sub
```

SubItemReasonWidth, DLRWndCmdTarget Property

Syntax SubItemReasonWidth= _Integer

Description This property indicates the width size in pixels of the Reason column within the DataLogger Window. A -1 value will be returned when the column is not displayed.

Parameter	Description
None	None

Result Integer

Example:

```
Dim objDLR As DLRWndCmdTarget
Public Sub Click()
        Debug.Print objDLR.SubItemReasonWidth
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetSynopticObject.GetSubObject("DLRWindow").GetObjectInterface
End Sub
```

SubItemTime, DLRWndCmdTarget Property

Syntax SubItemTime = _String

Description This property allows you set the text for the name of the "Time Event" column. When this field is left blank, the default text will be used instead.

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRWndCmdTarget
Public Sub Click()
        Debug.Print objDLR.SubItemTime
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetSynopticObject.GetSubObject("DLRWindow").GetObjectInterface
End Sub
```

SubItemTimePos, DLRWndCmdTarget Property

Syntax SubItemTimePos = _Integer

Description This property sets or returns the position of the "Time" column within the Data Logger window. When setting a new value, the other columns will be automatically re-positioned in the window layout. In addition when setting the "-1", the column will be hidden. The "0" value is used to indicate position of the first column on the left in the window.

Parameter	Description
None	None

Result Integer

Example:

```
Dim objDLR As DLRWndCmdTarget
Public Sub Click()
        Debug.Print objDLR.SubItemTimePos
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetSynopticObject.GetSubObject("DLRWindow").GetObjectInterface
End Sub
```

SubItemTimeWidth, DLRWndCmdTarget Property

Syntax SubItemTimeWidth = _Integer

Description This property indicates the width size in pixels of the Time column within the DataLogger Window. The -1 value will be returned if the column is not displayed.

Parameter	Description
None	None

Result Integer

Example:

```
Dim objDLR As DLRWndCmdTarget
Public Sub Click()
        Debug.Print objDLR.SubItemTimeWidth
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetSynopticObject.GetSubObject("DLRWindow").GetObjectInterface
End Sub
```

SubItemUser, DLRWndCmdTarget Property

Syntax SubItemUser = _String

Description This property allows you to set the text of the name to appear as the "User" Column's name. If left blank the default text will be used instead.

Parameter	Description
None	None

Result String

Example:

```
Dim objDLR As DLRWndCmdTarget
Public Sub Click()
        Debug.Print objDLR.SubItemUser
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetSynopticObject.GetSubObject("DLRWindow").GetObjectInterface
End Sub
```

SubItemUserPos, DLRWndCmdTarget Property

Syntax SubItemUserPos = _Integer

Description This property sets or returns the position of the "User" column within the DataLogger window. When setting a new value, the other columns will be automatically re-positioned in the window layout. In addition when setting the "-1", the column will be hidden. The "0" value is used to indicate position of the first column on the left in the window.

Parameter	Description
None	None

Result Integer

Example:

```
Dim objDLR As DLRWndCmdTarget
Public Sub Click()
        Debug.Print objDLR.SubItemUserPos
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetSynopticObject.GetSubObject("DLRWindow").GetObjectInterface
End Sub
```

SubItemUserWidth, DLRWndCmdTarget Property

Syntax SubItemUserWidth = _Integer

Description This property indicates the width size in pixels of the User column within the DataLogger Window. A -1 value will be returned if the column is not displayed.

Parameter	Description
None	None

Result Integer

Example:

```
Dim objDLR As DLRWndCmdTarget
Public Sub Click()
        Debug.Print objDLR.SubItemUserWidth
End Sub
Public Sub SymbolLoading()
        Set objDLR = GetSynopticObject.GetSubObject("DLRWindow").GetObjectInterface
End Sub
```

29. DrawCmdTarget

Click, Generic Event

Description Event occurs when the left or right mouse button is pressed within the design area.

Parameter	Description
None	None

DblClick, Generic Event

Description Event occurs when the right mouse key is double clicked within the design area. The double clicking time is set in operating system's settings.

Parameter	Description
None	None

KeyDown, Generic Event

Description Event occurs when a key is pressed down on the keyboard. This event returns the integer, KeyCode and Shift variables. This event is generated independently from being focused on.

Parameter	Description
KeyCode As Integer	Pressed Keys VBA Code. The VBA code is a set of constants which, in addition to the normal alphanumeric characters without lower/Uppercase distinction, also contemplates other keyboard keys such as the function keys, Caps Lock, etc.
Shift As Integer	Indices whether the Shift, Ctrl and Alt keys are pressed: 1 = SHIFT 2 = CTRL 4 = ALT

KeyPress, Generic Event

Description Event occurs when a key from the keyboards is pressed and released. This event returns the KeyAscii integer variable containing the pressed key's ASCII code. This event is generated only when the design is focused on.

Parameter	Description
Keyascii As Integ	The pressed key's ASCII code.

KeyUp, Generic Event

Description Event occurs when a key on the keyboard is released (after being pressed). This event releases the interger type keyCode and Shift variables. This event occurs indipendently of being focused on.

Parameter	Description
KeyCode As Integer	The pressed key's VBA code. The VBA code is a set of constants that, apart from the normal alphanumeric characters, without upper/lowercase distinction, contemplates other keys such as the Caps Lock function key etc.
Shift As Integer	Indicates whether whether the Shift, Ctrl and Alt keys are pressed: 1 = SHIFT 2 = CTRL 4 = ALT

KillFocus, Generic Event

Description Event occurs when the object in question is deselected or loses focus.

Parameter	Description
None	None

MouseDown, Generic Event

Description Event notified both in the screen code and in the object code every time the mouse key is clicked on screen, independently from its position or symbol. This event returns the integer Button and Shift type variables and the X and Y single type variables.
In order to manage this event only within a screen object you will need to use the "IsCursorOnObject" function.

Parameter	Description
Button As Integer	Indicates pressed mouse button: 1 = Left 2 = Right 4 = Central
Shift As Integer	Indicates whether the Shift, Ctrl and Alt keys are pressed: 1 = SHIFT 2 = CTRL 4 = ALT

X As Single	Horizontal coordinates referring to the cursor's position when event occurs.
Y As Single	Vertical coordinates referring to the cursor's position when event occurs.

MouseMove, Generic Event

Description Event notified both in the screen code and the object code when the mouse cursor changes position on screen, independently from the position or symbol. This event returns the Button and Shift integer type variables and the X and Y single type variables.
In order to manage this evenly only within a screen object you will need to use the "IsCursorOnObject" function.

Parameter	Description
Button As Integer	Pressed mouse key index: 1 = Left 2 = Right 4 = Central
Shift As Integer	Indicates whether the Shift, Ctrl and Alt keys are pressed: 1 = SHIFT 2 = CTRL 4 = ALT
X As Single	Horizontal coordinate referring to the cursor's positon when event occurs.
Y As Single	Vertical coordinate referring to the cursor's position when event occurs.

MouseUp, Generic Event

Description Event notified both in the screen and object codes when any one of the mouse keys are released on screen, independently from its position or symbol. This event returns the Button and Shift integer type variables and the X and Y single type variables.
In order to manage this event only within an object on screen you will need to use the "IsCursorOnObject" function.

Parameter	Description
Button As Integer	Pressed mouse key index: 1 = Left 2 = Right 4 = Central
Shift As Integer	Indicates whether the Shft, Ctrl and Alt keys are pressed: 1 = SHIFT 2 = CTRL 4 = ALT
X As Single	Horizontal coordinates referring to the cursor's positon when event occurs
Y As Single	Vertical coordinates referring to the cursor's position when event occurs

OnChange, Generic Event

Description Event occurs when the symbol object changes its graphic status. This event returns the index relating to which graphic event changed.

The indexes are returned as follows:
enum_ONCHANGE_COMPOSEDMOVE
enum_ONCHANGE_SCALE
enum_ONCHANGE_MOVEX
enum_ONCHANGE_MOVEY
enum_ONCHANGE_TITLE
enum_ONCHANGE_STARTINGX
enum_ONCHANGE_STARTINGY
enum_ONCHANGE_ENDINGX
enum_ONCHANGE_ENDINGY
enum_ONCHANGE_FILLING
enum_ONCHANGE_ROTATION
enum_ONCHANGE_VISIBLE
enum_ONCHANGE_EDGECOLOR
enum_ONCHANGE_BACKCOLOR
enum_ONCHANGE_FILLCOLOR
enum_ONCHANGE_BITMAP
enum_ONCHANGE_XROTATIONCENTER
enum_ONCHANGE_YROTATIONCENTER

Parameter	Description
ChangeType As Integer	Graphic index change

OnChangeExecutionCanceled, Generic Event

Description Event occurs for symbol objects which try to get or move the current synapses in execution.

Parameter	Description
None	None

OnChangeExecutionToPromoter, Generic Event

Description Event occurs for symbol objects executing sysnapses which have been notified that another object is trying to change their execution flow. Setting the bRet parameter to False will stop this from happening.

Parameter	Description
None	None

OnExecutionPending, Generic Event

Description Event occurs once every second according to the symbol object executing the synapsy by indicating that the system is waiting for this object to pass this execution to another object by means of the SynapseValue, SynapseValueFromID o SynapsePassExecution functions.

Parameter	Description
None	None

OnFireExecution, Generic Event

Description Event occurs each time the design object's synapsy is about to be executed. The system establishes which design to activate the synapsy based on the logic flow represented by connections and their tabulation order. The moment the output synapsy are set in the design object with its sysnapsy in execution, the system passes the macro execution on to another design by triggering the OnFireExecution event. Therefore the logic functions concerning the handling of the symbol's input and output synapsy are inserted in this event.

Parameter	Description
None	None

OnFireSynapse, Generic Event

Description This event is generated every time an input synapse receives the value from an output synapse linked to it, or in other words then a drawing object has set the value of the output synapse, by means of using the SynapseValue, SynapseValueFromID or SynapsePassExecution properties to which the input synapse, receiving the event, is linked.
You can find out which input synapse has been effected by the event described by using the SynapseName parameter.

Parameter	Description
SynapseName As String	Name of the synapse which has just received the value.

OnPostPaint, Generic Event

Description Event occurs each time the design object gets its graphics refreshed by the system. This can happen under different circumstances, for example following the opening of a screen, when the application is focused on, and every time the design's animation is executed or its position recalculated on screen.
The hde parameter (Handle to the device context) gieves useful information for the expert Windows user on the graphic refresh types adopted by Movicon.

Parameter	Description
ByVal hdc As Long	Handle to the device context.

OnPrePaint, Generic Event

Description Event occurs each time the design object gets its graphics refreshed by the system. This can happen under different circumstances, for example following the opening of a screen, when the application is focused on, and every time the design's animation is executed or its position recalculated on screen.

The symbol's graphic refresh is disabled when the bRet parameter is set to false.
The hde parameter (Handle to the device context) gieves useful information for the expert Windows user on the graphic refresh types adopted by Movicon.

Parameter	Description
ByVal hdc As Long	Handle to the device context
bRet As Boolean	Enable redesign

OnTextChanged, Generic Event

Description Event occurs when the text of the object's title has been changed with the keyboard. The ChangedText string varialbe containing the new text is returned.

In cases where the project's password managment has been enabled, the "OnTextChanged" event in objects will automatically request for user login in accordance to the password level set in that object.

Parameter	Description
ChangedText As String	New text containing the Title field.

OnTextChanging, Generic Event

Description Event occurs when the object's title is changed with the keyboard. The bRet boolean variable allows or does not allow this change: when the bRet variable is set to False within the function, the changes made to the text contained in the object will have no effect.

Parameter	Description

bRet As Boolean	Text changing enabling

OnTimer, Generic Event

Description Event occurs with a period of about 1/2 seconds (time not guaranteed) during runtime mode. During the Test mode this period is proportional to the set test velocity. The event's execution time can be customized by means of the TimerEventFrequence registry key.

Parameter	Description
None	None

OnToolTip, Generic Event

Description Event occurs when the obnject is in a condition to display a "pop-up" string called ToolTip (eg. pointed by mouse). the Show boolean variable is returned True when the mouse is on the object and False when the mouse is outside the object. To display th ToolTip you must set the relevant method.

Parameter	Description
Show As Boolean	Variable which indicates whether the mouse cursor is on the object.

SetFocus, Generic Event

Description Event occurs when the design object receives focus or is selected.

Parameter	Description
None	None

SymbolLoading, Generic Event

Description Event notified when the drawing object is loaded in memory, therefore at the opening of the screen it belongs to. This event is independent of the design's visibility conditions.

Parameter	Description
None	None

SymbolUnloading, Generic Event

Description Event occurs when the design object is unloaded from memory when the screen closes. This event is indiependent of the design's visibility conditions.

Parameter	Description
None	None

AddPolyPoint, DrawCmdTarget Function

Syntax AddPolyPoint(_nPos, _nX, _nY)

Description This function is used in polygon drawings for adding a new vertex. The point is put in the segment indicated with nPos keeping in mind that the segment from the first point to the second point drawn assumes the number 1, and the second segment from the second to the third point drawn, assumes the number 2 and so forth until the last (the last point to the first point) which assumes the 0 value. The new vertex will have Cartesian coordinates expressed in nX and nY pixels. An additional new point will cause the segment numeration to change. For example, let's take a polygon with 10 points (10 segments from 0 to 9) when adding a point in the seventh position the new segment will be numbered with 8, therefore the segment which was number 8 will become number 9 and the segment which was number 9 will become number 10.
The function will return with a False value when the position does not exist.

Parameter	Description
nPos As Integer	Segment position where a new point of the polygon is inserted in the center.
nX As Integer	X Coordinate in pixels from the screen's origin in which the new point is to be positioned.
nY As Integer	Y Coordinate in pixels from the screen's origin in which the new point is to be positioned.

Result Boolean

Example:

```
Public Sub Click()
        MsgBox(CStr(AddPolyPoint(0, 10, 10)),vbOkOnly,GetProjectTitle)
End Sub
```

CloseThisSynoptic, DrawCmdTarget Function

Syntax CloseThisSynoptic()

Description Closes the synoptic containing the object in which this function is called. This function has no effect when called in the startup screen. This function is used for closing the window (screen) opened in modal mode or in a separate frame.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        CloseThisSynoptic
End Sub
```

ConvertAngleToPoint, DrawCmdTarget Function

Syntax ConvertAngleToPoint(_nXOffset, _nYOffset, _nAngle, _nRadius, _pnX, _pnY)

Description This function allows you to identify the X, Y coordinates of the Angle's point of origin. The parameters passed with this function identify the angle (nAngle), the length (nRadius) and the offset in respect to the vector's start and end points (nXOffset and nYOffset).
This function writes the calculated coordinates in the pnX and pnY parameters.

Parameter	Description
nXOffset	X Offset.
nYOffset	Y Offset.
nAngle	Angle of vector in degrees.
nRadius	Length of vector.
pnX	Calculated X coordinate.
pnY	Calculated Y coordinate.

Result None

Example:

```
Public Sub Click()
        Dim nXOffset As Integer
        Dim nYOffset As Integer
        Dim nAngle As Double
        Dim nRadius As Double
        Dim pnX As Integer
        Dim pnY As Integer
```

```
        ' Vector offset
        nXOffset = -10
        nYOffset = 10
        ' Angle and length
        nAngle = 0
        nRadius = 100
        ' Convert to point
        ConvertAngleToPoint(nXOffset, nYOffset, nAngle, nRadius,pnX, pnY)
        ' View X,Y
        MsgBox "pnX=" & pnX & " pnY=" & pnY, vbOkOnly, "Test CenterRotation"
        'Return-> 90,10
End Sub
```

ConvertPointToAngle, DrawCmdTarget Function

Syntax ConvertPointToAngle(_nXOffset, _nYOffset, _nX, _nY)

Description This function permits you to identify the angle of the vector defined by the nX, nY origin coordinates and the offset in respect to the orgin nXOffset, nYOffset of origin. This function returns the horizontal angle's value of the vector from 0 to 360 degrees in anticlockwise.

Parameter	Description
nXOffset	X Offset.
nYOffset	Y Offset.
nX	X coordinate of origin point.
nY	Y coordinate of origin point.

Result Long

Example:

```
Public Sub Click()
        Dim nXOffset As Integer
        Dim nYOffset As Integer
        Dim nX As Integer
        Dim nY As Integer
        Dim nResult As Long

        ' Vector offset (Offset vettore)
        nXOffset = -10
        nYOffset = 10
        ' Vector orgin (Origine vettore)
        nX = 0
        nY = 0
        ' Calculate (Calcola)
        nResult = ConvertPointToAngle(nXOffset, nYOffset, nX, nY)
        MsgBox("Angle=" & nResult, ,"ConvertPointToAngle")
End Sub
```

CursorPosToObjectPos, DrawCmdTarget Function

Syntax CursorPosToObjectPos(_pnX, _pnY)

Description This functions sets and converts the cursor's pointer position to the position of the object. Accepts two Integer type parameters for the X and Y positions. This functions converts the cursor's position in function with the drawing belonging to a symbol by returning the original coordinates independently from any changes made to the symbol's sizses.

Parameter	Description
nX As Integer	X coordinate.
nY As Integer	Y coordinate.

Result None

Example:
If the object calling this code is not contained in a symbol the returned values will be identical.

```
Public Sub Click()
        Dim nX As Integer
        Dim nY As Integer
        GetCursorPos(nX, nY)
        MsgBox("X=" & nX & " Y=" & nY,vbOkOnly,GetProjectTitle)
        CursorPosToObjectPos(nX, nY)
        MsgBox("X=" & nX & " Y=" & nY,vbOkOnly,GetProjectTitle)
End Sub
```

DeletePolyPoint, DrawCmdTarget Function

Syntax DeletePolyPoint(_nPos)

Description This function is used in polygon drawings for cancelling vertex. The position is indicated with nPos by taking into consideration that the first drawing is set with the 0 value. This operation involves a number of segments and points. For example, if you have a polygon with 10 points (from 0 to 9) and you take away the 7th point, this will turn the 8th point into the 7th point and 9th point into the 8th point.
If the position does not exist no points will be taken away and the function will return with the False value.

Parameter	Description
nPos As Integer	Segment position to be deleted.

Result Boolean

Example:

```
Public Sub Click()
        MsgBox(CStr(DeletePolyPoint(0)),vbOkOnly,GetProjectTitle)
```

```
End Sub
```

EnableVariableEvent, DrawCmdTarget Function

Syntax EnableVariableEvent(_lpszVariableName, _bEnable)

Description This function enables or disables the execution of the event linked to the change of a specified variable (see On..[VARIABLE]..Changed). Accepts the string type lpszVariableName parameters containing the name of the variable responsible for the event and bEnable boolean type containing the event enabling.

Parameter	Description
lpszVariableName As String	Variable name
bEnable As Boolean	event enabling

Result Boolean

Example:

```
Public Sub Click()
        EnableVariableEvent("VAR00001",False)
End Sub
```

GetActiveXObject, DrawCmdTarget Function

Syntax GetActiveXObject()

Description This function returns an Object type parameter corresponding to an ActiveX inserted into a Movicon screen. It is used for accessing to the object's properties.

This function is not supported in Windows CE.(If set, always returns 'null')

Parameter	Description
None	None

Result Object

Example:

```
'Screen ambit
Public Sub Click()
        Dim obj As Object
        Dim app As Object
        Set obj =GetAbsoluteSubObject("ObjectName")
```

```
        Set app = obj .GetActiveXObject
        ' properties depend on the object type
        app.Visible = True  'Set the visibility of the ActiveX
        Set app =Nothing
        Set obj =Nothing
    ocx.Locked = True  'blocls the data changes inside the  combo boxEnd Sub
'Object ambit
Public Sub Click()
        Dim app As Object
        Set app = GetActiveXObject()
        ' properties depend on the object type
        app.Visible = True  'Set the visibility of the ActiveX
        Set app = Nothing
End Sub
```

GetAlias, DrawCmdTarget Function

Syntax GetAlias(_lpszAlias, _bRecursive)

Description This function returns the valued defined for the Alias passes as the "lpszAlias" parameter. The bRecursive parameter consents the Alias to be searched for in the ocal object table or also in the container symbol or in the screen.

Parameter	Description
lpszAlias As String	Name of Alias where to retrieve value.
bRecursive As Boolean	If set to True, this parameter consents Alias search in symbol contain and screen as well.

Result String

Example:

```
Public Sub Click()
        MsgBox "Alias <<TsetAlais>> = " & GetAlias("TsetAlais", False),vbInformation,
        GetProjectTitle
End Sub
```

GetAliasListName, DrawCmdTarget Function

Syntax GetAliasListName()

Description This function returns the list of Aliases defined in the object. A string will be returned where the name of the Aliases are separated by the "|" (pipe) character.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        MsgBox "Alias List = " & GetAliasListName(),vbInformation, GetProjectTitle
End Sub
```

GetAliasListValue, DrawCmdTarget Function

Syntax GetAliasListValue()

Description This function returns the list of values associated to the Aliases defined in the object. A string will be returned where the values of the Aliases are separated by the "|" (pipe) character.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        MsgBox "Alias Value List = " & GetAliasListValue(),vbInformation,
        GetProjectTitle
End Sub
```

GetCommandsInterfaceOnRelease, DrawCmdTarget Function

Syntax GetCommandsInterfaceOnRelease ()

Description This function gets the CommandsListCmdTarget interface relating to the object's command list. This interface can be used for modifying the referenced object's "Commands on Click" list.

Parameter	Description
None	None

Result Object: returns a CommandsListCmdTarget type object.

Example:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim objCommandList As CommandsListCmdTarget
```

```
        Set objRect = GetSynopticObject.GetSubObject("objRect")
        Set objCommandList = objRect.GetCommandsInterfaceOnRelease

        Set objCommandList = Nothing
        Set objRect = Nothing
End Sub
```

GetConnectorObjectConnected, DrawCmdTarget Function

Syntax GetConnectorObjectConnected(_lpszSynapseName, _nConnection)

Description This function allows you access the properties and methods of the connector object applied to the screen. The synapses in the drawing is identified by its name with the lspzSynapseName parameter while the connection number is defined in the nConnection parameter, given that more than one connector object can be connected to one synapses.

Parameter	Description
lpszSynapseName As String nConnection As Integer	Name of the synapses set in the object. Number of the connection to be considered.

Result Object

Example:

```
Public Sub OnFireExecution()
         Set edge color of symbol connected
        ' (Imposta il colore dei simboli collegati)
        Dim obj As Object

        For i = 0 To GetNumConnectionsOnSynapse("OUT") - 1
                Set obj = GetConnectorObjectConnected("OUT", i)
                obj.EdgeColor = lColor
                Set obj = Nothing
        Next i
End Sub
```

GetContainerObject, DrawCmdTarget Function

Syntax GetContainerObject()

Description This function returns a Object type parameter corresponding to the container symbol. It is used for accessing to the symbol's properties containing the drawing in question.

Parameter	Description

None	None

Result Object
If Function has been executed successfully it will retrieve an object of type DrawCmdTarget otherwise Nothing is returned.

Example:

```
Public Sub Click()
        Dim app As DrawCmdTarget
        Set app = GetContainerObject()
        If app Is Nothing Then
                MsgBox("app is Nothing",vbOkOnly,GetProjectTitle)
        Else
                MsgBox(app.Title,vbOkOnly,GetProjectTitle)
        End If
        Set app = Nothing
End Sub
```

GetCursorPos, DrawCmdTarget Function

Syntax GetCursorPos(_pnX, _pnY)

Description This function returns the current position of the cursor by means of two returned Integer type parameters. The coordinates are expressed in pixels and refer to the top left corner of the screen window.

Parameter	Description
nX As Integer	X Coordinate.
nY As Integer	Y Coordinate.

Result None

Example:

```
Public Sub Click()
        Dim nX As Integer
        Dim nY As Integer
        GetCursorPos (nX,nY)
        MsgBox("X=" & nX & " Y=" & nY,vbOkOnly,GetProjectTitle)
End Sub
```

GetCursorPosInObject, DrawCmdTarget Function

Syntax GetCursorPosInObject(_pnX, _pnY)

Description This function returns the current position of the cursor by means of two returned Integer type parameters. The coordinates are expressed in pixels and refer to the top left corner of the object.

Parameter	Description
nX As Integer	X Coordinate.
nY As Integer	Y Coordinate.

Result None

Example:

```
Public Sub Click()
        Dim nX As Integer
        Dim nY As Integer
        GetCursorPosInObject(nX,nY)
        MsgBox("X=" & nX & " Y=" & nY,vbOkOnly,GetProjectTitle)
End Sub
```

GetGaugeObject, DrawCmdTarget Function

Syntax GetGaugeObject()

Description This function permits you to access the specific properties and methods of a vectorial gauge object inserted on screen. The properties and methods mentioned are described in the GaugeCmdTarget interface.
The use of this function does not make it necessary to insert the vectorial gauge into a symbol as with the GetSubGaugeObject function.

Parameter	Description
None	None

Result Object
If Function has been executed successfully it will retrieve an object of type GaugeCmdTarget otherwise Nothing is returned.

Example:

```
Public Sub Click()
        Dim ObjGauge As GaugeCmdTarget
        Set ObjGauge = GetSynopticObject.GetSubObject("Gauge").GetGaugeObject
        If ObjGauge Is Nothing Then MsgBox "ObjGauge Is Nothing", vbExclamation + vbOkOnly,
        "GetGaugeObject"
        Set ObjGauge = Nothing
End Sub
```

GetNumConnectionsOnSynapse, DrawCmdTarget Function

Syntax GetNumConnectionsOnSynapse(_lpszSynapseName)

Description This function lets you establish the number of connector objects are linked to the synapses, by identifying them with the name through the lpszSynpseName parameter.

Parameter	Description
lpszSynapseName As String	Name of the synapses set in the object.

Result Integer

Example:

```
Public Sub OnFireExecution()
        ' Set edge color of connector
        Dim obj As Object

        For i = 0 To GetNumConnectionsOnSynapse("OUT") - 1
         Set obj = GetConnectorObjectConnected("OUT", i)
         obj.EdgeColor = lColor
         Set obj = Nothing
        Next i
End Sub
```

GetNumPolyPoint, DrawCmdTarget Function

Syntax GetNumPolyPoint()

Description This function is used in polygon drawings for retrieving the number of segments (points) which compose the polygon specified.

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        MsgBox("Numero di segmenti: " & GetNumPolyPoints,vbOkOnly,GetProjectTitle)
End Sub
```

GetNumSynapsis, DrawCmdTarget Function

Syntax GetNumSynapsis()

Description This function allows you verify the number of synapses containing the drawing object.

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        MsgBox("GetNumSynapsis  =" & GetNumSynapsis,vbOkOnly,GetProjectTitle)
End Sub
```

GetObjectConnectedOnSynapse, DrawCmdTarget Function

Syntax GetObjectConnectedOnSynapse(_lpszSynapseName, _nConnection)

Description This function allows you to access the properties and methods of a drawing object connected to the synapse through the connector object. The synapse in the drawing is identified with its name through the lpszSynpseName parameter while the number of the connection is defined in the nConnection parameter as more than one connector object can be linked to one synapses.

Parameter	Description
lpszSynapseName As String	Name of the synapses set in the object.
nConnection As Integer	Connection number.

Result Object

Example:

```
Public Sub OnFireExecution()
        ' Set edge color of symbol connected
        Dim obj As Object

        For i = 0 To GetNumConnectionsOnSynapse("OUT") - 1
         Set obj = GetObjectConnectedOnSynapse("OUT", i)
         obj.EdgeColor = lColor
         Set obj = Nothing
        Next i
End Sub
```

GetObjectInterface, DrawCmdTarget Function

Syntax GetObjectInterface()

Description This function allows you to access the properties and methods related to some of the objects inserted on screen. The properties and methods are those described in this manual in the relative chapters. The objects whose properties can be accessed are:

Chart	ChartWndCmdTarget
Trend	TrendCmdTarget
Gauge	GaugeCmdTarget
AlarmWindow	AlarmWndCmdTarget

DataLoggerWindow DLRWndCmdTarget
HistoryLog Window HisLogWndCmdTarget

Parameter	Description
None	None

Result Object

Example:

```
'Object on screen'
Public Sub Click()
        Dim objTrend As TrendCmdTarget
        Set objTrend = GetSynopticObject.GetSubObject("Trend").GetObjectInterface
        Debug.Print objTrend.Samples
        Set objTrend = Nothing
End Sub
'Object in a symbol on screen'
Public Sub Click()
        Dim objTrend As TrendCmdTarget
        Set objTrend = GetContainerObject.GetSubObject("Trend").GetObjectInterface
        Debug.Print objTrend.Samples
        Set objTrend = Nothing
End Sub
```

GetOnScreenPosition, DrawCmdTarget Function

Syntax GetOnScreenPosition(_pnLeft, _pnTop, _pnRight, _pnBottom)

Description This function returns the position of the object on screen by means of a four Long type parameters indicating the coordinates in pixels of the object's left, top, right and bottom.

Parameter	Description
pnLeft As Long	Left coordinate
pnTop As Long	Top coordinate
pnRight As Long	Right coordinate
pnBottom As Long	Bottom coordinate

Result None

Example:

```
Public Sub Click()
        Dim nLeft As Long
        Dim nTop As Long
        Dim nRight As Long
        Dim nBottom As Long
```

```
        GetOnScreenPosition(nLeft , nTop, nRight, nBottom)
        MsgBox("Left =" & nLeft & " Top =" & nTop & " Right =" & nRight & " Bottom =" & nBottom
        ,vbOkOnly,GetProjectTitle)
End Sub
```

GetPolyPointOnScreenX, DrawCmdTarget Function

Syntax GetPolyPointOnScreenX(_nPos)

Description This function is used in polygon drawings for retrieving the X coordinate of a polygon point. The position is indicated in the nPos taking into consideration that the first point has the 0 value. The coordinates are expressed in pixels and always refers to the screen's origin coordinate.

Parameter	Description
nPos As Integer	Point's position

Result Integer

Example:

```
Public Sub Click()
        MsgBox("Coordinata X: " & GetPolyPointOnScreenX(0),vbOkOnly,GetProjectTitle)
End Sub
```

GetPolyPointOnScreenY, DrawCmdTarget Function

Syntax GetPolyPointOnScreenY(_nPos)

Description This function is used in polygon drawings for getting the Y coordinates of a polygon point. The nPos is used for indicating the position taking into consideration that the first point has the 0 value. The coordinates are expressed in pixels and always refer to the screen's origin coordinate.

Parameter	Description
nPos As Integer	Point position

Result Integer

Example:

```
Public Sub Click()
        MsgBox("Coordinata X: " & GetPolyPointOnScreenY(0),vbOkOnly,GetProjectTitle)
End Sub
```

GetSubGaugeObject, DrawCmdTarget Function

Syntax GetSubGaugeObject()

Description Allows you to access the properties and methods of a vectorial gauge object inserted on screen. The properties and methods are those described in this manual in the GaugeCmdTarget chapter.

Parameter	Description
None	None

Result Object
If Function has been executed successfully it will retrieve an object of type GaugeCmdTarget otherwise Nothing is returned.

Example:

```
Public Sub Click()
        Dim ObjGauge As GaugeCmdTarget
        Set ObjGauge = GetContainerObject.GetSubGaugeObject("Gauge")
        If ObjGauge Is Nothing Then MsgBox "ObjGauge Is Nothing", vbExclamation + vbOkOnly,
        "GetSubGaugeObject"
        Set ObjGauge = Nothing
End Sub
```

GetSubObject, DrawCmdTarget Function

Syntax GetSubObject(_lpszObjectName)

Description This function allows you to reference an object contained in the symbol's internal. This simply means that you get access to a drawing,which is one of the various drawings making one symbol and which is more complex than the symbol itself.
This function can also be used by an "Embedded Screen or "Tab Group" object for retrieving an object contained within them.

Parameter	Description
lpszObjectName As String	Object Name.

Result Object
If Function has been executed successfully it will retrieve an object of type DrawCmdTarget otherwise Nothing is returned.

Example:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget

        Set objRect = GetSubObject("Rect1")
        objRect.BackColor = objRect.BackColor + 10
        Set objRect = Nothing
End Sub
```

Example 2:

```
Dim objSyn As SynopticCmdTarget
Dim objEmbeddedScreen As DrawCmdTarget
Dim objDraw As DrawCmdTarget

Public Sub Click()
        Set objSyn = GetSynopticObject
        Set objEmbeddedScreen = objSyn.GetSubObject("oEmbeddedScreen")
        Set objDraw = objEmbeddedScreen.GetSubObject("objRect1")

        If (objDraw.BackColor <> vbRed) Then
        objDraw.BackColor = vbRed
        Else
        objDraw.BackColor = vbWhite
        End If

        Set objSyn = Nothing
        Set objEmbeddedScreen = Nothing
        Set objDraw = Nothing
End Sub
```

GetSubTrendObject, DrawCmdTarget Function

Syntax GetSubTrendObject()

Description Permits you to access the properties and methods of an on screen trend object. The properties and methods are those described in this manual in the TrendCmdTarget chapter.

Parameter	Description
None	None

Result Object
If Function has been executed successfully it will retrieve an object of type TrendCmdTarget otherwise Nothing is returned.

Example:

```
Public Sub Click()
        Dim ObjTrend As TrendCmdTarget
        Set ObjTrend = GetContainerObject.GetSubTrendObject("Trend")
        If ObjTrend Is Nothing Then MsgBox "ObjTrend Is Nothing", vbExclamation + vbOkOnly,
        "GetSubTrendObject"
        Set ObjTrend = Nothing
End Sub
```

GetSynapseName, DrawCmdTarget Function

Syntax GetSynapseName(_nID)

Description This function returns the name of the synapse by returning a nID parameter with the synapse's progressive entry number.

Parameter	Description
nID As Integer	Synapse's progressive entry number.

Result String

Example:

```
Public Sub Click()
        Dim Result As Integer
        Result = GetNumSynapsis
        If Resul > 0 Then
                MsgBox("GetSynapseName =" & GetSynapseName(0),vbOkOnly,GetProjectTitle)
        end if
End Sub
```

GetSynapsePoint, DrawCmdTarget Function

Syntax GetSynapsePoint(_lpszSynapseName, _pnX, _pnY)

Description This function is used in polygon drawings for getting the X coordinates of the polygon's points. The position is indicated with the nPos keeping in mind that the first point has the 0 value. The coordinates are expressed in pixel and always refer to the screen's origin coordinate.

Parameter	Description
lpszSynapseName As String	Name of the synapse set in the object.
nX As Integer	Horizontal position from object's left side.
nY As Integer	Horizontal position from the object's right side.

Result Boolean

Example:

```
Public Sub Click()
        Dim nX As Integer
        Dim nY As Integer

        ' Get synpase point
        GetSynapsePoint("IN", nX, nY)
        ' View result
        MsgBox("X="& nX & " Y=" & nY, , "GetSynapsePoint")
End Sub
```

GetSynopticObject, DrawCmdTarget Function

Syntax GetSynopticObject()

Description Allows access to the properties and methods of the synoptic containing the object. The properties and the methods are those described in this manual in the SynopticCmdTarget section.

Parameter	Description
None	None

Result Object
If Function has been executed successfully it will retrieve an object of type SynopticCmdTarget otherwise Nothing is returned.

Example:

```
Public Sub Click()
        Dim app As Object
        Dim obj2 As Object 'Object 2

        Set app = GetSynopticObject()
        If app Is Nothing Then
        MsgBox("app is Nothing",vbOkOnly,GetProjectTitle)
        Else
        Set obj2 = app.GetSubObject("Object 2")
        MsgBox(obj2 .Title,vbOkOnly,GetProjectTitle)
        Set obj2 = Nothing
        End If
        Set app = Nothing
End Sub
```

GetTrendObject, DrawCmdTarget Function

Syntax GetTrendObject()

Description This function allows access to the properties and methods of an on screen trend object. The properties and methods are those described in this manual in the TrendCmdTarget section.
When this function is used, unlike the GetSubTrendObject, it is not necessary to insert the trend into a symbol.

Parameter	Description
None	None

Result Object
If Function has been executed successfully it will retrieve an object of type TrendCmdTarget otherwise Nothing is returned.

Example:

```
Public Sub Click()
        Dim ObjTrend As TrendCmdTarget
        Set ObjTrend = GetSynopticObject.GetSubObject("Trend").GetTrendObject
        If ObjTrend Is Nothing Then MsgBox "ObjTrend Is Nothing", vbExclamation + vbOkOnly,
        "GetTrendObject"
        Set ObjTrend = Nothing
End Sub
```

GetUniqueObjectID, DrawCmdTarget Function

Syntax GetUniqueObjectID()

Description This function returns the ID number automatically associated by Movicon to a drawing containing synapses.
The 0 returned value indicates that the drawing object does not contain any set synapses.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        MsgBox("Numero di ID: " & GetUniqueObjectID,vbOkOnly,GetProjectTitle)
End Sub
```

GetXMLSettings, DrawCmdTarget Function

Syntax GetXMLSettings()

Description This function returns the definition string of the object in the project in XML project.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        MsgBox("GetXMLSettings =" & GetXMLSettings,vbOkOnly,GetProjectTitle)
End Sub
```

HasSynapsis, DrawCmdTarget Function

Syntax HasSynapsis()

Description This function identified whether the object contains synapses.
The true returned value indicates the presence of at least one synapses in the symbol otherwise the False value will be returned.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        If (HasSynapsis()) Then
                MsgBox "HasSynapsis = " & CStr(HasSynapsis()), vbOkOnly, "Test HasSynapsis"
        End If
End Sub
```

InflateObject, DrawCmdTarget Function

Syntax InflateObject(_nCX, _nCY)

Description Changes the object's size in function with the two parameters: the nCX parameter for the proportions on X axis and the nCy for the proportiohns on the Y axis. The size will change on both sides of the symbol according to the number of pixels set in the parameters (eg. if nX = 5 the size will increase by 5 pixels on the right and 5 pixels on the left).

Parameter	Description
nCX As Integer	Proportion on X axis.
nCY As Integer	Proportion on Y axis.

Result None

Example:

```
Public Sub Click()
        InflateObject(5,5)
        MsgBox "Width=" & width & " Height=" & height & " Xpos=" & Xpos & " Ypos=" & Ypos,
        vbOkOnly, GetProjectTitle
End Sub
```

IsCursorOnObject, DrawCmdTarget Function

Syntax IsCursorOnObject()

Description This function controls whether the mouse pointer is on the object. When it is on the object a True value will be returned otherwise a False value will be returned.

Parameter	Description

None	None

Result Boolean

Example:

```
Public Sub MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
        If (IsCursorOnObject()) Then
        MsgBox "IsCursorOnObject = " & CStr(IsCursorOnObject()), vbOkOnly, "Test
        IsCursorOnObject "
        End If
End Sub
```

IsGlobalObjectName, DrawCmdTarget Function

Syntax IsGlobalObjectName()

Description This function verifies whethe the object's name is set as global.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        MsgBox("IsGlobalObjectName: " & CStr(IsGlobalObjectName), vbOkOnly,GetProjectTitle)
End Sub
```

IsSynapseConnected, DrawCmdTarget Function

Syntax IsSynapseConnected(_lpszSynapseName)

Description This function allows you to verify whether a synapses is linked to a connector object by identifying its name through the IspzSynapseName parameter.
The function returns the true value when the synapse is linked to a connector object.

Parameter	Description
lpszSynapseName As String	Name of the synapse set in the object.

Result Boolean

Example:

```
Public Sub Click()
        Dim Result As Boolean
        Result = IsSynapseConnected("IN")
        MsgBox("IsSynapseConnected =" & Result,vbOkOnly , "GetSynapsePoint")
End Sub
```

LoadExtSettings, DrawCmdTarget Function

Syntax LoadExtSettings

Description This function allows you to load the object configuration form the relative external configuration file. This file can be specified in the "Configuration File" property while in design mode, or by using the in the "ExtSettingsFile" interface property. ".SXML" is the extension provided for this file.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
Dim objSymbol As DrawCmdTarget
Set objSymbol = GetSynopticObject.GetSubObject("TestObject")
If objSymbol Is Nothing Then Exit Sub
objSymbol.ExtSettingsFile = "test.sxml"
objSymbol.LoadExtSettings
Set objSymbol = Nothing
End Sub
```

MoveObject, DrawCmdTarget Function

Syntax MoveObject(_nLeft, _nTop, _nRight, _nBottom)

Description This function moves and resizes the object in function with the parameters set in pixels as integer values. The parameters required are Left (left side position), Top (top position), Right (right side position) and Bottom (bottom position). All the parameters always refer to the screen's top left corner.

Parameter	Description
nLeft As Integer	Top left corner X coordinate value.
nTop As Integer	Top left corner Y coordinate value.

nRight As Integer	Bottom right corner X coordinate value.
nBottom As Integer	Bottom right corner Y coordinate value.

Result None

Example:

```
Public Sub Click()
        MoveObject(100,200,330,450)
        MsgBox "Width=" & width & " Height=" & height & " Xpos=" & Xpos & " Ypos=" & Ypos,
        vbOkOnly, GetProjectTitle
End Sub
```

OffsetObject, DrawCmdTarget Function

Syntax OffsetObject(_nX, _nY)

Description Causes the object to move in the X and Y directions by the number of pixels set in the nX integer parameter for the X direction and nY parameter for the Y direction. The positive offset in the X direction is towards the bottom of the screen and towards the right for the Y direction. This function sets the offset new values and does not return any values.

Parameter	Description
nX As Integer	Movement along the X axis value.
nY As Integer	Movement along the Y axis value.

Result None

Example:

```
Public Sub Click()
        OffsetObject(100,200)
        MsgBox "Width=" & width & " Height=" & height & " Xpos=" & Xpos & " Ypos=" & Ypos,
        vbOkOnly, GetProjectTitle
End Sub
```

PolyPointX, DrawCmdTarget Function

Syntax PolyPointX(_nPos)

Description This property is used in polygon drawings for getting the X coordinate of a point or to change it. The nPos indicates the position considering that the first point drawn is position 0. The coordinates are expressed in pixels and refer to the drawing's container: screen or symbol.

Parameter	Description

nPos As Integer	Point position (0 corresponds to the first point drawn).

Result Integer

Example:

```
Public Sub Click()
        MsgBox "PolyPointX = " & PolyPointX(0) , vbOkOnly, GetProjectTitle
End Sub
```

PolyPointY, DrawCmdTarget Function

Syntax PolyPointY(_nPos)

Description This property is used in polygon drawings for getting the Y coordinate of a point or to change it. The nPos indicates the position considering that the first point drawn is position 0. The coordinates are expressed in pixels and refer to the drawing's container: screen or symbol.

Parameter	Description
nPos As Integer	Point Position (0 corresponds to the first point drawn).

Result Integer

Example:

```
Public Sub Click()
        MsgBox "PolyPointY= " & PolyPointY(0) , vbOkOnly, GetProjectTitle
End Sub
```

PrintThisSynoptic, DrawCmdTarget Function

Syntax PrintThisSynoptic(_nMode, _bKeepPrintProportions)

Description Prints the screen containing the symbol. Accepts the nMode parameter indicating the print mode.

The parameter can have the following the values:
0=select printer
1=direct printout
2=Preview

This property is not supported in Windows CE.

Parameter	Description
nMode As Integer	Print mode.
bKeepPrintProportions as boolean	Optional Parameter. When set at "True" permits prints with the same proportions

	as seen on screen. When set at 'False' (default value) both height and width are adapted to fit within page sizes.

Result None

Example:

```
'Screen Environment
Public Sub Click()
        PrintThisSynoptic(2)
End Sub
```

Prop, DrawCmdTarget Function

Syntax Prop(_lpzPropName)

Description Sets or returns the value of a property setup in the drawing with the lpszPropName. The new properties are persistent, which means their values remain intact after being saved and after the project has been closed and system shut down. By using the OnCustomizeSymbol event in the templates library you can customize the insertion of symbols on screen and create and set properties with the Prop function, which are then used in read in the remaining template codes.

Parameter	Description
lpzPropName As String	Name of property.

Result String

Example:

```
'Let's suppose we have a screen with two symbols representing two buttons called 'P1' and 'P2'.
In the 'P1' symbol we will write the following code:
Public Sub Click()
        If Prop("Premuto") = "ON" Then
        Prop("Premuto") = "OFF"
        Else
        Prop("Premuto") = "ON"
        End If
End Sub
'In the 'P2' symbol we will test the 'Pressed' property of 'P1' in the following way:
Dim syn As Object
Dim rett As Object
Public Sub SymbolLoading()
        Set syn = GetSynopticObject
        Set rett = syn.GetSubObject("P1")
End Sub
Public Sub SymbolUnloading()
        Set syn = Nothing
        Set rett = Nothing
End Sub
Public Sub Click()
        Debug.Print rett.Prop("Premuto")
End Sub
```

RemoveAlias, DrawCmdTarget Function

Syntax RemoveAlias(_lpszAlias)

Description This function lets you remove the Alias passed as the "lpszAlias" parameter.

Parameter	Description
lpszAlias As String	Name of Alias to be removed.

Result Boolean

Example:

```
Public Sub Click()
        Debug.Print RemoveAlias("TsetAlais")
End Sub
```

RemoveAllAliases, DrawCmdTarget Function

Syntax RemoveAllAliases()

Description This function removes all the Aliases defined in the object's Table.

Parameter	Description
None	None

Result None

Example:

```
Public Sub Click()
        RemoveAllAliases()
End Sub
```

ResetColors, DrawCmdTarget Function

Syntax ResetColors()

Description This function resets the colors in the object. The settings include the contrast, brightness, greyness scale, background and line/text colors.

Parameter	Description

None	None

Result None

Example:

```
Public Sub Click()
        BackColorBrightness =200
        lRet = BackColorBrightness
        MsgBox "BackColorBrightness = " & CStr(lRet), vbOkOnly, GetProjectTitle
        ResetColors
        MsgBox "ResetColors hes been done", vbOkOnly, GetProjectTitle
End Sub
```

SaveExtSettings, DrawCmdTarget Function

Syntax SaveExtSettings

Description This function allows the object's configuration to be saved in the relative external configuration file. This file can be specified in the "Configuration File" properties while in design mode or by means of using the "ExtSettingsFile" interface property. The extension provided for this file is ".SXML".

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
Dim objSymbol As DrawCmdTarget
Set objSymbol = GetSynopticObject.GetSubObject("TestObject")
If objSymbol Is Nothing Then Exit Sub
objSymbol.ExtSettingsFile = "test.sxml"
objSymbol.SaveExtSettings
Set objSymbol = Nothing
End Sub
```

ScaleObject, DrawCmdTarget Function

Syntax ScaleObject(_lpar)

Description This method enlarges or reduces the object's size according to the percentage specified in the integer parameter. For example, the value 150 will enlarge the object one and half times its actual size, the value 50 will reduce the object of half its actual size.

Parameter	Description

lpar As Integer	Enlarge/Reduce value.

Result None

Example:
Public Sub Click()
    ScaleObject(80)
End Sub

SetAlias, DrawCmdTarget Function

Syntax SetAlias(_lpszAlias, _lpszValue)

Description This function sets the value defined for the Alias passed as the "lpszAlias" parameter. The new value will be the one passed with the "lpszValue " parameter and may be a variable name or a string or numeric value.
If the Alias does not exist in the object's Table it will be added as a new one.

After having added or modified a command from the object's command list you must execute the SaveChanges method from the CommandsListCmdTarget interface to apply modifications to the object's command list.
Please also remember that any modifications to command lists will only remain valid until the object is downloaded from memory (closing screen). The object will be restored with the initial command list associated when programmed the next time it is uploaded. However, command list modifications can be made persistent by associating the object with a configuration file which must be saved after modifying and saving the object's command list.

Parameter	Description
lpszAlias As String	Name of the Alias for which the value is set. If Alias does not exist, it will be added as a new one.
lpszValue As String	Value to set the Alias with.

Result Boolean

Example:
Public Sub Click()
    Debug.Print **SetAlias**("TsetAlais", "VAR00001")
End Sub

ShowPropList, DrawCmdTarget Function

Syntax ShowPropList()

Description This function shows a list of properties, created with the "Prop" function, and its corresponding values if boolean referable.

Parameter	Description
None	None

Result None

Example:

```
'Screen Environment
Public Sub Click()
        ShowPropList
End Sub
```

SynapseBackColor, DrawCmdTarget Property

Syntax SynapseBackColor(_lpszSynapseName)

Description This property sets or returns the back color for the synapse specified in the drawing object, by identifying the name through the lpszSynapseName parameter.The synapses are represented on the drawing by a colored circle and its position is set when the synapse is being entered.
The set or returned value contains the back color code (R,G,B, on each byte). You may find it more useful to use the RGB function for identifying the color easier.

Parameter	Description
lpszSynapseName As String	Synapse's name.

Result Long

Example:

```
Public Sub OnFireExecution()
        Dim lColor As Long
        If SynapseValue("OUT") = True Then
        lColor = RGB(0, 255, 0) ' Green color (Colore verde)
        Else
        lColor = RGB(255, 0, 0) ' Red color (Colore rosso)
        End If

        ' Set color (Imposta il colore)
        SynapseBackColor("OUT") = lColor
End Sub
```

SynapsePassExecution, DrawCmdTarget Function

Syntax SynapsePassExecution(_nID)

Description This function passes on the macro execution to the next drawing object. This operation is normally done by using the SynapseValue or the SynapseValueFromID functions.
This method is handy for "blind" objects, being objects which only have input synapses and need to pass on executions but do not have any output synapses to do so.

Parameter	Description
None	None

Result None

Example:

```
Public Sub OnFireExecution ()
        ' Continue the logic execution
        SynapsePassExecution
End Sub
```

SynapseValueFromID, DrawCmdTarget Property

Syntax SynapseValueFromID(_nID)

Description This property sets or returns the value of a synapse set in the object by identifying its ID number through the nID parameter.
The set or returned value is variant type and allows synapse compatibility with all data types set with the basic script language.

Parameter	Description
nID As Integer	Synapse's ID.

Result Variant

Example:

```
Public Sub OnFireExecution()
        Dim Result As Variant

        ' Read the old value (Legge il valore vecchio)
        Result = GetVariableValue("Temperature")
        ' Set value (Imposta il valore)
        SynapseValueFromID(0) = Result
End Sub
```

ZOrderMoveBack, DrawCmdTarget Function

Syntax ZOrderMoveToBack()

Description This function changes the object's order position on screen to one move back.

Parameter	Description
None	None

Result Boolean

Example:
Public Sub Click()
ZOrderMoveBack
End Sub

ZOrderMoveForward, DrawCmdTarget Function

Syntax ZOrderMoveForward()

Description This function changes the object's order position on screen to one move forward.

Parameter	Description
None	None

Result Boolean

Example:
Public Sub Click()
ZOrderMoveForward
End Sub

ZOrderMoveToBack, DrawCmdTarget Function

Syntax ZOrderMoveToBack()

Description This function changes the object's order position by moving it to the back being underneath all the other symbols on screen.

Parameter	Description
None	None

Result Boolean

Example:
Public Sub Click()
ZOrderMoveToBack
End Sub

ZOrderMoveToFront, DrawCmdTarget Function

Syntax ZOrderMoveToFront()

Description This function changes the order position of the object on screen by moving it to the front, being on top of all the other symbols.

Parameter	Description
None	None

Result Boolean

Example:
Public Sub Click()
ZOrderMoveToFront
End Sub

AdaptFontSize, DrawCmdTarget Property

Syntax AdaptFontSize = Boolean

Description This property sets or returns the text's font size adaption to the size of the object.

Parameter	Description
None	None

Result Boolean

Example:
Public Sub Click()
AdaptFontSize = Not **AdaptFontSize**

```
        MsgBox "AdaptFontSize = " & CStr(AdaptFontSize), vbOkOnly, "Test AdaptFontSize"
End Sub
```

AlignFont, DrawCmdTarget Property

Syntax AlignFont = Integer

Description This property sets or returns the text alignment in the object according to the Center, Right, Left, top, Bottom, center-left, center-right options available in the drawing style properties.
In addition, you can also use a "_movicon.efontFormat" type Enum. to put the chosen alignment type into effect:

enum_fft_center = 0 (in the center)
enum_fft_top = 1 (at the top)
enum_fft_bottom = 2 (at the bottom)
enum_fft_left = 3 (on the left)
enum_fft_right = 4 (on the right)
enum_fft_centerleft = 5 (center-left)
enum_fft_centerright = 6 (center-right)

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        Dim iRet As Integer
        AlignFont = enum_fft_center
        iRet = AlignFont
        MsgBox "AlignFont = " & CStr(iRet), vbOkOnly, "Text AlignFont"
        AlignFont = enum_fft_top
        iRet = AlignFont
        MsgBox "AlignFont = " & CStr(iRet), vbOkOnly, "Text AlignFont"
        AlignFont = enum_fft_bottom
        iRet = AlignFont
        MsgBox "AlignFont = " & CStr(iRet), vbOkOnly, "Text AlignFont"
        AlignFont = enum_fft_left
        iRet = AlignFont
        MsgBox "AlignFont = " & CStr(iRet), vbOkOnly, "Text AlignFont"
        AlignFont = enum_fft_right
        iRet = AlignFont
        MsgBox "AlignFont = " & CStr(iRet), vbOkOnly, "Text AlignFont"
        AlignFont = enum_fft_centerleft
        iRet = AlignFont
        MsgBox "AlignFont = " & CStr(iRet), vbOkOnly, "Text AlignFont"
        AlignFont = enum_fft_centerright
        iRet = AlignFont
        MsgBox "AlignFont = " & CStr(iRet), vbOkOnly, "Text AlignFont"
End Sub
```

AlignFontOffsetX, DrawCmdTarget Property

Syntax AlignFontOffsetX= _Long

Description This property sets or returns the "AlignFontOffsetX" property value indicating the 'X' position Offset in pixels of the text set in an object's Title properties. The offset moves title according to the object's "Text Align Font" property settings.

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim sRet As Long

        sRet = InputBox "Insert Offset X"

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        objRect.AlignFontOffsetX = sRet

        MsgBox "AlignFontOffsetX= " & objRect.AlignFontOffsetX, vbOkOnly,
        GetProjectTite
        Set objRect = Nothing
End Sub
```

AlignFontOffsetY, DrawCmdTarget Property

Syntax AlignFontOffsety= _Long

Description This property sets or returns the "AlignFontOffsetY" property value indicating the 'Y' position Offset in pixels of the text set in an object's Title properties. The offset moves title according to the object's "Text Align Font" property settings.

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim sRet As Long
```

```
        sRet = InputBox "Insert Offset y"

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        objRect.AlignFontOffsety = sRet

        MsgBox "AlignFontOffsety= " & objRect.AlignFontOffsety, vbOkOnly,
        GetProjectTite
        Set objRect = Nothing
End Sub
```

AutoRepeatClick, DrawCmdTarget Property

Syntax AutoRepeatClick = _Integer

Description This property (zero for default) when set allows you to enter the execution frequency of the click event while the user keeps the mouse key pressed down on the symbol. This frequency is the multiple of the OnTimer event execution frequency. Accepts Integer values.

Parameter	Description
lNone	Click event frequency.

Result Integer

Example:

```
Public Sub SymbolLoading()
        AutoRepeatClick = 1
End Sub
Dim nCounter As Integer
Public Sub Click()
        nCounter = nCounter + 1
        Debug.Print "Varore di conteggio = " & CStr(nCounter )
End Sub
```

BackBrushPattern, DrawCmdTarget Property

Syntax BackBrushPattern = Integer

Description This property sets or returns the object's background style according to the options also offered in the Brush Style property from the Fill attributes properties section.
Values 0 to 22 can be used which group a series of back hatched styles (vertical lines, points, bricks, fabrics, etc.). Value -1 (FFFF esadecimal) eliminates the applied back style with the set 'Solid' option.
There are three object back patterns:

1 =Transparent
2 = Solid
3 = Pattern

To set the back with solid or transparent you must use the "BackBrushVisible" function. The back pattern selection can then be done only when the back is not set with the "transparent" option.

This property is not supported in Windows CE.(only allowed set at -1)

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        Dim iRet As Integer
        BackBrushPattern = 0
        iRet = BackBrushPattern
        MsgBox "BackBrushPattern = " & CStr(iRet), vbOkOnly, "Test BackBrushPattern"
        BackBrushPattern = &HFFFF
        iRet = BackBrushPattern
        MsgBox "BackBrushPattern = " & CStr(iRet), vbOkOnly, "Test BackBrushPattern"
        BackBrushPattern = 22
        iRet = BackBrushPattern
        MsgBox "BackBrushPattern = " & CStr(iRet), vbOkOnly, "Test BackBrushPattern"
End Sub
```

BackBrushVisible, DrawCmdTarget Property

Syntax BackBrushVisible = Boolean

Description Sets or returns the condition of the drawing's back brush visibility. When set at True the back brush pattern is visible otherwise it will be transparent. When the back is set with the "solid" option you can then change the pattern with the "BackBrushPattern".

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        BackBrushVisible = Not BackBrushVisible
        MsgBox "BackBrushVisible = " & CStr(BackBrushVisible), vbOkOnly, "Test
        BackBrushVisible"
End Sub
```

BackColor, DrawCmdTarget Property

Syntax BackColor = Long

Description This property sets or returns the object's back colour.
The passed r returned value contains the back colour code (R,G,B, in each byte). The RGB function is handy to use for an easier way to identify the colour.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim lRet As Long
        BackColor = RGB(255,0,0)
        lRet = BackColor
        MsgBox "BackColor = " & CStr(lRet), vbOkOnly, "Test BackColor"
        BackColor = RGB(0,255,0)
        lRet = BackColor
        MsgBox "BackColor = " & CStr(lRet), vbOkOnly, "Test BackColor"
        BackColor = RGB(0,0,255)
        lRet = BackColor
        MsgBox "BackColor = " & CStr(lRet), vbOkOnly, "Test BackColor"
End Sub
```

BackColorBrightness, DrawCmdTarget Property

Syntax BackColorBrightness = _Integer

Description This property sets or returns the Brightness component associated to the drawing's back colour. Each colour is represented by a long value which can be sub-divided into three whole components: hue, saturation and brightness. Values from 0 to 255 can be used in this property.

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        Dim lRet As Integer
        BackColorBrightness =185
        lRet = BackColorBrightness
        MsgBox "BackColorBrightness = " & CStr(lRet), vbOkOnly, "Test BackColorBrightness "
        BackColorBrightness = 192
        lRet = BackColorBrightness
        MsgBox "BackColorBrightness = " & CStr(lRet), vbOkOnly, "Test BackColorBrightness "
        BackColorBrightness =200
        lRet = BackColorBrightness
        MsgBox "BackColorBrightness = " & CStr(lRet), vbOkOnly, "Test BackColorBrightness "
End Sub
```

BackColorHue, DrawCmdTarget Property

Syntax BackColorHue = _Integer

Description This property sets or returns the Hue component associated to the drawing's back colour. Each colour is represented by a long value which can be sub-divided into three whole components: hue, saturation and brightness. Values from 0 to 255 can be used in this property.

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        Dim lRet As Integer
        BackColorHue =185
        lRet = BackColorHue
        MsgBox "BackColor = " & CStr(lRet), vbOkOnly, "Test BackColorHue "
        BackColorHue = 192
        lRet = BackColorHue
        MsgBox "BackColorHue = " & CStr(lRet), vbOkOnly, "Test BackColorHue "
        BackColor =200
        lRet = BackColorHue
        MsgBox "BackColorHue = " & CStr(lRet), vbOkOnly, "Test BackColorHue "
End Sub
```

BackColorSaturation, DrawCmdTarget Property

Syntax BackColorSaturation = _Integer

Description This property sets or returns the Saturation component associated to the drawing's back colour. Each colour is represented by a long value which can be sub-divided into three whole components: hue, saturation and brightness. Values from 0 to 255 can be used in this property.

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        Dim lRet As Integer
        BackColorSaturation =185
```

```
        lRet = BackColorSaturation
        MsgBox "BackColorSaturation = " & CStr(lRet), vbOkOnly, "Test BackColorSaturation "
        BackColorSaturation = 192
        lRet = BackColorSaturation
        MsgBox "BackColorSaturation = " & CStr(lRet), vbOkOnly, "Test BackColorSaturation "
        BackColorSaturation =200
        lRet = BackColorSaturation
        MsgBox "BackColorSaturation = " & CStr(lRet), vbOkOnly, "Test BackColorSaturation "
End Sub
```

BitmapAlignment, DrawCmdTarget Property

Syntax BitmapAlignment= eImageAligns

Description This property sets or returns the alignment type index of the image set as the object's background Static Image. This alignment type can be specified using the eImageAligns enumerator or by inserting the corresponding numeric value:

enum_ima_stretch = 0 (Stretched)
enum_ima_topleft = 1 (Top-Left)
enum_ima_topcenter = 2 (Top-Center)
enum_ima_topright = 3 (Top-Right)
enum_ima_centerleft = 4 (Center-Left)
enum_ima_center = 5 (Center)
enum_ima_centerright = 6 (Center-Right)
enum_ima_bottomleft = 7 (Bottom-Left)
enum_ima_bottomcenter = 8 (Bottom-Center)
enum_ima_bottomright = 9 (Bottom-Right)

Setting this property to the value 0 (Stretched) will automatically set the "BitmapStretched" property to True and viceversa.

Parameter	Description
None	None

Result eImageAligns

Example:

```
Public Sub Click()

        Select Case BitmapAlignment
        Case enum_ima_stretch
        MsgBox "BitmapAlignment = Stretched(" & CStr(BitmapAlignment) & ")"
        Case enum_ima_topleft
        MsgBox "BitmapAlignment = TopLeft(" & CStr(BitmapAlignment) & ")"
        Case enum_ima_topcenter
        MsgBox "BitmapAlignment = TopCenter(" & CStr(BitmapAlignment) & ")"
        Case enum_ima_topright
        MsgBox "BitmapAlignment = TopRight(" & CStr(BitmapAlignment) & ")"
        Case enum_ima_centerleft
        MsgBox "BitmapAlignment = CenterLeft(" & CStr(BitmapAlignment) & ")"
        Case enum_ima_center
        MsgBox "BitmapAlignment = Center(" & CStr(BitmapAlignment) & ")"
        Case enum_ima_centerright
        MsgBox "BitmapAlignment = CenterRight(" & CStr(BitmapAlignment) & ")"
        Case enum_ima_bottomleft
```

```
        MsgBox "BitmapAlignment = BottomLeft(" & CStr(BitmapAlignment) & ")"
        Case enum_ima_bottomcenter
        MsgBox "BitmapAlignment = BottomCenter(" & CStr(BitmapAlignment) & ")"
        Case enum_ima_bottomright
        MsgBox "BitmapAlignment = BottomRight(" & CStr(BitmapAlignment) & ")"
        End Select

End Sub
```

BitmapID, DrawCmdTarget Property

Syntax BitmapID = _String

Description This property allows you to get or set the image to be shown with the object. The passed or returned string corresponds to the name of the image file.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Debug.Print BitmapID
End Sub
```

BitmapOffsetX, DrawCmdTarget Property

Syntax BitmapOffsetx= _Integer

Description This property sets or returns the X position in pixels of the image displayed as Static Image. The offset moves the image according to the object's "Image Alignment" property settings.

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim sRet As Integer

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        sRet = objRect.BitmapOffsetX
```

```
        MsgBox "BitmapOffsetX= " & sRet, vbOkOnly, GetProjectTite
        Set objRect = Nothing
End Sub
```

BitmapOffsetY, DrawCmdTarget Property

Syntax BitmapOffsety= _Integer

Description This property sets or returns the Y position in pixels of the image displayed as Static Image. The offset moves the image according to the object's "Image Alignment" property settings.

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim sRet As Integer

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        sRet = objRect.BitmapOffsetY

        MsgBox "BitmapOffsetY= " & sRet, vbOkOnly, GetProjectTite
        Set objRect = Nothing
End Sub
```

BitmapStretched, DrawCmdTarget Property

Syntax BitmapStretched = _Boolean

Description This property sets or returns the enabling to stretch the image set as the Static image in object's Background attributes properties.
This property always changed to True in conjunction with the "BitmapAlignment" property when set to the value 0 (stretched) and viceversa

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
```

```
        Dim objRect As DrawCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        objRect.BitmapStretched = Not(objRect.BitmapStretched)

        Set objRect = Nothing
End Sub
```

BitmapTransparent, DrawCmdTarget Property

Syntax BitmapTransparent = Boolean

Description This property activates or deactivated the Transparent property for the eventual image displayed in the object.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        BitmapTransparent = Not BitmapTransparent
        MsgBox "BitmapTransparent = " & CStr(BitmapTransparent), vbOkOnly, "Test
        BitmapTransparent "
End Sub
```

BitmapTransparentColor, DrawCmdTarget Property

Syntax BitmapTransparentColor = _Long

Description This property sets or returns the desired colour to be made transparent in the eventual image displayed in the drawing object.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        BitmapTransparentColor = RGB(255,255,255)
        MsgBox "BitmapTransparentColor  = " & CStr(BitmapTransparentColor), vbOkOnly,
        GetProjectTitle 'Return-> 16777215
End Sub
```

BitmapKeepAspectRatio, DrawCmdTarget Property

Syntax BitmapStretched = _Boolean

Description This property allows you to keep the original aspect ratio when image is stretched. This property can also be read.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget

        Set objRect = GetSynopticObject.GetSubObject("objRect")
        objRect.BitmapKeepAspectRatio = Not(objRect.BitmapKeepAspectRatio)

        Set objRect = Nothing
End Sub
```

BorderType, DrawCmdTarget Property

Syntax BorderType = _Integer

Description This property sets or returns the set border type for the object according the none, sunken, etched, bump or raised options which are also available from the symbol's general properties.
The following values can be used: 0=none, 1=bump, 2=etched, 3=raised, 4=sunken.

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        For i = 0 To 4 Step 1
        BorderType = i
        sRet = BorderType
        MsgBox "BorderType = " & sRet, vbOkOnly, GetProjectTitle
        Next i
End Sub
```

CenterRotation, DrawCmdTarget Property

Syntax CenterRotation = Boolean

Description This property sets or returns the enabling of the object's baricenter rotation in the center of the object itself.

This property is not support in Windows CE.(If set, always returns 'true')

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        CenterRotation = Not CenterRotation
        MsgBox "CenterRotation = " & CStr(CenterRotation ), vbOkOnly, "Test CenterRotation"
End Sub
```

DefStructName, DrawCmdTarget Property

Syntax DefStructName = _String

Description This property sets or returns the structure type variable set for the object. A structure type variable can be associated to each drawing or symbol and the member variables can be identified omitting the name of the structure variable. For example; ":HighLevel" instead of "Pump:HighLevel".

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sRet As String
        Dim lRet As Variant
        DefStructName= "STRUCT1"
        sRet = DefStructName
        MsgBox "DefStructName = " & sRet, vbOkOnly, GetProjectTitle
        sRet = sRet & ":VAR00001"
        lRet = GetVariableValue(sRet)
        MsgBox "VAR00001 Value = " & lRet, vbOkOnly, GetProjectTitle
End Sub
```

DefStructNameAbsolute, DrawCmdTarget Property

Syntax DefStructNameAbsolute = _String

Description This property sets or returns the structure type variable set for the object containing the drawing. Therefore this property is valid for a drawing contained within a symbol.
A structure type variable can be associated to each drawing or symbol and the member variables can be identified within the symbol/object by omitting the name of the structure variable. For example; ":HighLevel" instead of "Pump:HighLevel".

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sRet As String
        Dim lRet As Variant
        DefStructNameAbsolute = "STRUCT1"
        sRet = DefStructNameAbsolute
        MsgBox "DefStructNameAbsolute = " & sRet, vbOkOnly, GetProjectTitle
        sRet = sRet & ":VAR00001"
        lRet = GetVariableValue(sRet)
        MsgBox "VAR00001 Value = " & lRet, vbOkOnly, GetProjectTitle
End Sub
```

DrawingState, DrawCmdTarget Property

Syntax DrawingState = _Integer

Description This property sets or returns the display type associated to the drawing's back color. The values which can be used are from 0 to 3 which correspond to the following settings: 0=Normal, 1=SemiTransparent, 2=Disabled and 3=Dither.

This property is not supported in Windows CE.(If set, always returns -1)

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        Dim lRet as Integer
        DrawingState =0
        lRet = DrawingState
        MsgBox "DrawingState = " & CStr(lRet), vbOkOnly, "Test DrawingState "
        DrawingState =1
        lRet = DrawingState
        MsgBox "DrawingState = " & CStr(lRet), vbOkOnly, "Test DrawingState "
        DrawingState =2
        lRet = DrawingState
        MsgBox "DrawingState = " & CStr(lRet), vbOkOnly, "Test DrawingState "
        DrawingState =3
        lRet = DrawingState
        MsgBox "DrawingState = " & CStr(lRet), vbOkOnly, "Test DrawingState "
End Sub
```

DrawingStateShadow, DrawCmdTarget Property

Syntax DrawingStateShadow = _Integer

Description This property sets or returns the display type associated to the back colour of the drawing's shadow. The Values from 0 to 3 can be used which correspond to the following settings: 0=Normal, 1=SemiTransparent, 2=Disabled and 3=Dither. This property can be used when the Shadow property is set at True.

This property is not supported in Windows CE.(If set, always returns -1)

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        Dim lRet as Integer
        DrawingStateShadow =0
        lRet = DrawingStateShadow
        MsgBox "DrawingStateShadow = " & CStr(lRet), vbOkOnly, "Test DrawingStateShadow "
        DrawingStateShadow =1
        lRet = DrawingStateShadow
        MsgBox "DrawingStateShadow = " & CStr(lRet), vbOkOnly, "Test DrawingStateShadow"
        DrawingStateShadow =2
        lRet = DrawingStateShadow
        MsgBox "DrawingStateShadow = " & CStr(lRet), vbOkOnly, "Test DrawingStateShadow"
        DrawingStateShadow =3
        lRet = DrawingStateShadow
        MsgBox "DrawingStateShadow = " & CStr(lRet), vbOkOnly, "Test DrawingStateShadow"
End Sub
```

EdgeColor, DrawCmdTarget Property

Syntax EdgeColor = Long

Description This property sets or returns the colour of the object's edge.
The values passed or returned contain the edge color code (R,G,B, in each byte). A handy and easier way to identify the colour would be to use the RBG function.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim lRet As Long
        EdgeColor = RGB(255,0,0)
        lRet = EdgeColor
        MsgBox "EdgeColor = " & CStr(lRet), vbOkOnly, "Test EdgeColor"
        EdgeColor = RGB(0,255,0)
        lRet = EdgeColor
        MsgBox "EdgeColor = " & CStr(lRet), vbOkOnly, "Test EdgeColor"
        EdgeColor = RGB(0,0,255)
        lRet = EdgeColor
        MsgBox "EdgeColor = " & CStr(lRet), vbOkOnly, "Test EdgeColor"
End Sub
```

EmbeddedSynoptic, DrawCmdTarget Property

Syntax EmbeddedSynoptic = _String

Description This property sets or returns the name of the screen displayed within the embedded screen object. Therefore this property is valid for this type of object only. When used in other drawing types an error will be generated.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        MsgBox "EmbeddedSynoptic = " & EmbeddedSynoptic , vbOkOnly, GetProjectTitle
End Sub
```

EnableExecution, DrawCmdTarget Property

Syntax EnableExecution = _Boolean

Description This property sets or returns the enabling of the execution property management. When this is set at False the processing of all the configured graphic functions will be inhibited. Alerts of basic events generated by variable value changes will also be inhibited.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        EnableExecution = Not EnableExecution
        MsgBox "EnableExecution = " & EnableExecution, vbOkOnly, GetProjectTitle
End Sub
```

EnableVariable, DrawCmdTarget Property

Syntax EnableVariable = _String

Description This property sets or returns the name of the referenced object's enable variable. Inserting this variable will enable the component when the variable's value is different to zero.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim sVarName As String

        GetVariableNameFromList(sVarName)
        Set objRect = GetSynopticObject.GetSubObject("objRect")
        objRect.EnableVariable = sVarName

        Set objRect = Nothing
End Sub
```

ExtSettingsFile, DrawCmdTarget Property

Syntax ExtSettingsFile = _String

Description This property sets or returns the external configuration file for the referenced object. This file can also be specified in the object's "Configuration File" property in Design mode. The extension provided for this file is ".SXML".

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
Dim objSymbol As DrawCmdTarget
Set objSymbol = GetSynopticObject.GetSubObject("TestObject")
If objSymbol Is Nothing Then Exit Sub
objSymbol.ExtSettingsFile = "test.sxml"
objSymbol.SaveExtSettings
Set objSymbol= Nothing
End Sub
```

FillBrushPattern, DrawCmdTarget Property

Syntax FillBrushPattern = Integer

Description This property sets or returns the back brush pattern of the gradient filling of the object when the "Filling' animation property has been enabled.
Values between 0 and 22 can be used which group a series of back hatched styles (vertical lines, points, clothe, bricks, etc.,) which can be viewed in the Brush Styles property from the Fill Attributes section. The applied back style is eliminated when the value -1 (FFFF) is used.

This property is not completely supported in Windows CE (accepts the -1 value only).

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        Dim iRet As Integer
        FillBrushPattern = 0
        iRet = FillBrushPattern
        MsgBox "FillBrushPattern = " & CStr(iRet), vbOkOnly, "Test FillBrushPattern"
```

```
        FillBrushPattern = &HFFFF
        iRet = FillBrushPattern
        MsgBox "FillBrushPattern = " & CStr(iRet), vbOkOnly, "Test FillBrushPattern"
        FillBrushPattern = 22
        iRet = FillBrushPattern
        MsgBox "FillBrushPattern = " & CStr(iRet), vbOkOnly, "Test FillBrushPattern"
End Sub
```

FillColor, DrawCmdTarget Property

Syntax FillColor = Long

Description This property sets or returns the object's Fill colour when the Filling property has been enabled in the Animations properties section and when other colours have not been set through the "Variable Color" property or any thresholds edited.
The value passed or returned contains the fill color code (R,G,B, in each byte). It may be handier and easier to use the RGB function to identify the color.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim lRet As Long
        FillColor = RGB(255,0,0)
        lRet = FillColor
        MsgBox "FillColor = " & CStr(lRet), vbOkOnly, "Test FillColor"
        FillColor = RGB(0,255,0)
        lRet = FillColor
        MsgBox "FillColor = " & CStr(lRet), vbOkOnly, "Test FillColor"
        FillColor = RGB(0,0,255)
        lRet = FillColor
        MsgBox "FillColor = " & CStr(lRet), vbOkOnly, "Test FillColor"
End Sub
```

FillingMode, DrawCmdTarget Property

Syntax FillingMode = Integer

Description This property sets or returns the type of object's filling direction. Accepts an integer parameter containing the value (1,2,3,4) for the direction of the filling.

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        Dim iRet As Integer
        FillingMode = 1
        iRet = FillingMode
        MsgBox "FillingMode = " & CStr(iRet), vbOkOnly, "Test FillingMode"
End Sub
```

FillingPercent, DrawCmdTarget Property

Syntax FillingPercent = Integer

Description This property sets or returns the value of the object's filling percent. Accepts an interger parameter containing the object's filling value. Note: the filling value expressed in percentages must be between 0 and 100.

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        Dim iRet As Integer
        FillingPercent = 30
        iRet = FillingPercent
        MsgBox "FillingPercent = " & CStr(iRet), vbOkOnly, "Test FillingPercent"
        FillingPercent = 50
        iRet = FillingPercent
        MsgBox "FillingPercent = " & CStr(iRet), vbOkOnly, "Test FillingPercent"
        FillingPercent = 80
        iRet = FillingPercent
        MsgBox "FillingPercent = " & CStr(iRet), vbOkOnly, "Test FillingPercent"
End Sub
```

Font3D, DrawCmdTarget Property

Syntax Font3D = Integer

Description This property sets or returns the three-dimensional effect of the text in the object.

3D effect styles:
0=none
1=raised
2=embossed

Parameter	Description

None	None

Result Integer

Example:

```
Public Sub Click()
        Dim iRet As Integer
        font3D = 2
        iRet = font3D
        MsgBox "Font3D = " & CStr(iRet), vbOkOnly, "Test Font3D"
        font3D = 1
        iRet = font3D
        MsgBox "Font3D = " & CStr(iRet), vbOkOnly, "Test Font3D"
        font3D = 0
        iRet = font3D
        MsgBox "Font3D = " & CStr(iRet), vbOkOnly, "Test Font3D"
End Sub
```

FontBold, DrawCmdTarget Property

Syntax FontBold = Boolean

Description This property sets or returns the selected style's title font in Bold.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        FontBold = Not FontBold
        MsgBox "FontBold = " & CStr(FontBold ), vbOkOnly, GetProjectTitle
End Sub
```

FontCharSet, DrawCmdTarget Property

Syntax FontCharSet = _Integer

Description This property allows you to change the character set of the drawing's font by means of using a basic code. The usable values taken from the Microsoft SDK platform are listed below.

The CharSet can be:
0 = ANSI_CHARSET
1 = DEFAULT_CHARSET
2 = SYMBOL_CHARSET
128 = SHIFTJIS_CHARSET
129 = HANGEUL_CHARSET

129 = HANGUL_CHARSET
134 = GB2312_CHARSET
136 = CHINESEBIG5_CHARSET
255 = OEM_CHARSET

For WinNT 4.0 only:

77 = MAC_CHARSET
130 = JOHAB_CHARSET
177 = HEBREW_CHARSET
178 = ARABIC_CHARSET
161 = GREEK_CHARSET
162 = TURKISH_CHARSET
163 = VIETNAMESE_CHARSET
186 = BALTIC_CHARSET
222 = THAI_CHARSET
238 = EASTEUROPE_CHARSET
204 = RUSSIAN_CHARSET

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        FontCharSet = 2
        Debug.Print "Varore di charset = " & CStr(FontCharSet)
End Sub
```

FontEscapement, DrawCmdTarget Property

Syntax FontEscapement = _Integer

Description This property sets or returns the font escapement for the text in the object.
Values between 0 and 359 can be used which represent the rotation degrees of the title within the drawing. This property is available in the Windows 32/64 bit only.

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        If i < 359 Then
        FontEscapement = FontEscapement + 10
        sRet = FontEscapement
        MsgBox "FontEscapement = " & sRet, vbOkOnly, GetProjectTitle
        Else
        i = 0
```

```
        End If
End Sub
```

FontHeight, DrawCmdTarget Property

Syntax FontHeight = Integer

Description This property sets or returns the font's size in pixels for the text within the object. Accepts an Integer parameter.

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        Dim iRet As Integer
        FontHeight = 10
        iRet = FontHeight
        MsgBox "FontHeight = " & CStr(iRet), vbOkOnly, "Test FontHeight"
        FontHeight = 20
        iRet = FontHeight
        MsgBox "FontHeight = " & CStr(iRet), vbOkOnly, "Test FontHeight"
End Sub
```

FontItalic, DrawCmdTarget Property

Syntax FontItalic = Boolean

Description This property sets or returns the Italic style selected for the font.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        FontItalic = Not FontItalic
        MsgBox "FontItalic = " & CStr(FontItalic ), vbOkOnly, GetProjectTitle
End Sub
```

FontName, DrawCmdTarget Property

Syntax FontName = String

Description This property sets or returns the name of the font for text within the object. Accepts a String parameter.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sRet As String
        FontName = "Arial"
        sRet = FontName
        MsgBox "FontName = " & sRet, vbOkOnly, "Test FontName"
        FontName = "Times New Roman"
        sRet = FontName
        MsgBox "FontName = " & sRet, vbOkOnly, "Test FontName"
End Sub
```

GradientColor, DrawCmdTarget Property

Syntax GradientColor = _Long

Description This property sets or returns the object's Gradient color. Accepts a Long type parameter containing the colour's RGB code (R,G,B, on each byte). You may find it more useful to use the Movicon RGB function.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        GradientColor =RGB(255,255,255)
        sRet = GradientColor
        MsgBox "GradientColor = " & sRet, vbOkOnly, GetProjectTitle 'Return-> 16777215
End Sub
```

GradientFill, DrawCmdTarget Property

Syntax GradientFill = _Integer

Description This property sets or returns a numeric code corresponding to the direction of the gradient color. The different values are:
0 = none
1 = from right to left
2 = from centre to the borders horizontally
3 = from left to right
4 = from bottom to top
5 = from the centre to the borders vertically
6 = from top to bottom
7 = from the center outwards to the borders
8 = from top left corner diagonally
9 = from top right corner diagonally
10 = from bottom right corner diagonally
11 = from bottom left corner diagonally

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        For i = 0 To 11 Step 1
        GradientFill = i
        sRet = GradientFill
        MsgBox "GradientFill = " & sRet, vbOkOnly, GetProjectTitle
        Next i
End Sub
```

Height, DrawCmdTarget Property

Syntax Height = _Long

Description This property returns or set the value assigned to the object's height.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim lRet As Long
        Height = Height + 10
        lRet = Height
        MsgBox "Height = " & CStr(lRet), vbOkOnly, GetProjectTitle
End Sub
```

Hilite, DrawCmdTarget Property

Syntax Hilite = _Boolean

Description This property causes inversion between the object's back color and its edge color.
This property can be used for highlighting a drawing's movement in a certain area of the screen. The true value enables the color inversion.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim bRet As Boolean
        Hilite = Not Hilite
        bRet = Hilite
        MsgBox "Hilite = " & CStr(bRet), vbOkOnly, "Test Hilite "
End Sub
```

LineArrowHeight, DrawCmdTarget Property

Syntax LineArrowHeight = _Integer

Description This property sets or returns the size of the arrows displayed in the drawing object. This property is valid only for line and connector objects, being the only ones which can shown arrows at the sides end.

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        LineArrowType =1
        LineArrowHeight = 5
        lRet = LineArrowHeight
        MsgBox "LineArrowHeight = " & CStr(lRet), vbOkOnly, "Test LineArrowHeight "
        LineArrowHeight = 12
        lRet = LineArrowHeight
        MsgBox "LineArrowHeight = " & CStr(lRet), vbOkOnly, "Test LineArrowHeight "
        LineArrowHeight = 30
        lRet = LineArrowHeight
        MsgBox "LineArrowHeight = " & CStr(lRet), vbOkOnly, "Test LineArrowHeight "
        LineArrowHeight = 25
        lRet = LineArrowHeight
        MsgBox "LineArrowHeight = " & CStr(lRet), vbOkOnly, "Test LineArrowHeight "
End Sub
```

LineArrowType, DrawCmdTarget Property

Syntax LineArrowType = _Integer

Description This porperty sets or returns the displayed arrow type in the drawing object. This property is valid only for line and connector objects, being the only ones which can shown arrows at the sides end. The values to be used are between 0 to 3 which correspond to the following settings: 0 = --
1 = <--
2 = -->
3 = <-->

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        LineArrowType =0
        lRet = LineArrowType
        MsgBox "LineArrowType = " & CStr(lRet), vbOkOnly, "Test LineArrowType "
        LineArrowType =1
        lRet = LineArrowType
        MsgBox "LineArrowType = " & CStr(lRet), vbOkOnly, "Test LineArrowType "
        LineArrowType =2
        lRet = LineArrowType
        MsgBox "LineArrowType = " & CStr(lRet), vbOkOnly, "Test LineArrowType "
        LineArrowType =3
        lRet = LineArrowType
        MsgBox "LineArrowType = " & CStr(lRet), vbOkOnly, "Test LineArrowType "
End Sub
```

LineEndingX, DrawCmdTarget Property

Syntax LineEndingX = _Long

Description This function sets or returns the value of the ending point on the Line object's X axis. Accepts a Long value. Note that the starting and ending line point settings establishes the direction used by the mouse for drawing the line.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
```

```
        Dim lRet As Long
        LineEndingX = LineEndingX + 1
        lRet = LineEndingX
        MsgBox "LineEndingX = " & CStr(lRet), vbOkOnly, GetProjectTitle
End Sub
```

LineEndingY, DrawCmdTarget Property

Syntax LineEndingY = _Long

Description This function sets or returns the value of the ending point on the Line object's Y axis. Accepts a Long value. Note that the starting and ending line point settings establishes the direction used by the mouse for drawing the line.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim lRet As Long
        LineEndingY = LineEndingY + 1
        lRet = LineEndingY
        MsgBox "LineEndingY = " & CStr(lRet), vbOkOnly, GetProjectTitle
End Sub
```

LineStartingX, DrawCmdTarget Property

Syntax LineStartingX = Long

Description This function sets or returns the value of the starting point on the Line object's X axis. Accepts a Long value. Note that the starting and ending line point settings establishes the direction used by the mouse for drawing the line.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim lRet As Long
        LineStartingX = LineStartingX + 1
        lRet = LineStartingX
        MsgBox "LineStartingX = " & CStr(lRet), vbOkOnly, "Test LineStartingX"
End Sub
```

LineStartingY, DrawCmdTarget Property

Syntax LineStartingY = _Long

Description This function sets or returns the value of the starting point on the Line object's Y axis. Accepts a Long value. Note that the starting and ending line point settings establishes the direction used by the mouse for drawing the line.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim lRet As Long
        LineStartingX = LineStartingY + 1
        lRet = LineStartingY
        MsgBox "LineStartingY = " & CStr(lRet), vbOkOnly, GetProjectTitle
End Sub
```

LinkedTextFormat, DrawCmdTarget Property

Syntax LinkedTextFormat = _String

Description The execution property of a drawing or symbol called 'Text' is composed of two entry fields: the variable and the format. The last one follows the syntax shown below:

Integer display
x
xx
xxx
xxxx
xxxxx

Decimal display
x.x
x.xx
x.xxx
x.xxxx
x.xxxxx

String type variables do not need to be set with a format.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        VariableLinkedText = InputBox("Variable name:", "Change variable",VariableLinkedText)
        LinkedTextFormat = InputBox("Format:","Change format string",LinkedTextFormat)
End Sub
```

LinkedTextFormatVariable, DrawCmdTarget Property

Syntax LinkedTextFormatVariable = String

Description This property returns or sets the name of the variable whose value will be used for determining the associated animation text variable's display format.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim objRectangle As DrawCmdTarget
        Set objRectangle = GetSynopticObject.GetSubObject("Rect1")
        MsgBox  "Rect1 Format Variable = " &
        objRectangle.LinkedTextFormatVariable, vbInformation,GetProjectTitle
        Set objRectangle = Nothing
End Sub
```

Look3D, DrawCmdTarget Property

Syntax Look3D = Boolean

Description This property sets or returns the drawings 3D look.

This property is not supported in Windows CE.(If used always returns false)

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Look3D = Not Look3D
        MsgBox "Look3D = " & CStr(Look3D ), vbOkOnly, "Test Look3D "
End Sub
```

Look3DPressed, DrawCmdTarget Property

Syntax Look3DPressed = Boolean

Description This property sets or returns the pressed display of the drawing with a 3D Look.

This property is not supported in Windows CE.(if used, always returns false)

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Look3D = True
        Look3DPressed = Not Look3DPressed
        MsgBox "Look3DPressed = " & CStr(Look3DPressed ), vbOkOnly, "Test Look3DPressed "
End Sub
```

MetaFile, DrawCmdTarget Property

Syntax MetaFile = _String

Description This property allows you to get or set the image to be shown with the object. The returned or passed string corresponds to the name of the image file.

This property is not supported in Windows CE.(if set, always returns an empty string)

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Debug.Print MetaFile
End Sub
```

MouseCapture, DrawCmdTarget Property

Syntax MouseCapture = Boolean

Description This property enables the notification of events linked to the mouse also when the cursor goes out of the screen. Accepts a Boolean value. False is the default value for this property.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        MouseCapture = Not MouseCapture
        MsgBox "MouseCapture = " & MouseCapture , vbOkOnly, GetProjectTitle
End Sub
```

ObjectName, DrawCmdTarget Property

Syntax ObjectName

Description This property returns (read only) the name assigned to the object through its general properties.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        MsgBox "ObjectName = " & ObjectName, vbOkOnly, "Test ObjectName"
End Sub
```

ObjectPublicName, DrawCmdTarget Property

Syntax ObjectPublicName = _String

Description This property sets or returns the referenced object's public name. Modifying this property in the object's dropping code the public name will be saved in the object's xml code automatically updating the object

according to the project's settings.
During the runtime phase the public name can only be changed in the object's "OnPreLoadingSymbol" event. This means that the symbol will be updated only when the property is set within the OnPreLoadingSymbol" event. This property will have not effect when set outside the OnPreLoadingSymbol" event.

Parameter	Description
None	None

Result String

Example:

```
Public Sub OnPreSymbolLoading()
        ObjectPublicName = "ReferenceSymbol"
End Sub
```

PenColorBrightness, DrawCmdTarget Property

Syntax PenColorBrightness = _Integer

Description This property sets or returns the Brightness component associated to the drawing's line/test color. Each color is represented by a long value which can be subdivided into three components: hue, saturation and brightness. The values from 0 to 255 can be used in this property.

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        Dim lRet As Integer
        PenColorBrightness =185
        lRet = PenColorBrightness
        MsgBox "PenColorBrightness = " & CStr(lRet), vbOkOnly, "Test PenColorBrightness "
        PenColorBrightness = 192
        lRet = PenColorBrightness
        MsgBox "PenColorBrightness = " & CStr(lRet), vbOkOnly, "Test PenColorBrightness "
        PenColorBrightness =200
        lRet = PenColorBrightness
        MsgBox "PenColorBrightness = " & CStr(lRet), vbOkOnly, "Test PenColorBrightness "
End Sub
```

PenColorHue, DrawCmdTarget Property

Syntax PenColorHue = _Integer

Description This property sets and returns the Hue component associated to the drawing's line/text. Each color is represented by a long value which can be subdivided into three components: hue, saturation and brightness. The values from 0 to 255 can be used in this property.

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        Dim lRet As Integer
        PenColorHue =185
        lRet = PenColorHue
        MsgBox "PenColorHue = " & CStr(lRet), vbOkOnly, "Test PenColorHue "
        PenColorHue = 192
        lRet = PenColorHue
        MsgBox "PenColorHue = " & CStr(lRet), vbOkOnly, "Test PenColorHue "
        PenColorHue =200
        lRet = PenColorHue
        MsgBox "PenColorHue = " & CStr(lRet), vbOkOnly, "Test PenColorHue "
End Sub
```

PenColorSaturation, DrawCmdTarget Property

Syntax PenColorSaturation = _Integer

Description This property sets and returns the Saturation component associated to the drawing's line/text. Each color is represented by a long value which can be subdivided into three components: hue, saturation and brightness. The values from 0 to 255 can be used in this property.

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        Dim lRet As Integer
        PenColorSaturation =185
        lRet = PenColorSaturation
        MsgBox "PenColorSaturation = " & CStr(lRet), vbOkOnly, "Test PenColorSaturation  "
        PenColorSaturation = 192
        lRet = PenColorSaturation
        MsgBox "PenColorSaturation = " & CStr(lRet), vbOkOnly, "Test PenColorSaturation "
        PenColorSaturation =200
```

```
        lRet = PenColorSaturation
        MsgBox "PenColorSaturation = " & CStr(lRet), vbOkOnly, "Test PenColorSaturation "
End Sub
```

PenStyle, DrawCmdTarget Property

Syntax PenStyle = Integer

Description Sets or returns the style of the drawing's border line. (solid, dash, dot, dash-dot, dash-dot-dot, null). In order to activate this property the line width must be equal to 1, which is set through Pen Size property in the Stroke Attributes properties section or with the PenWidth method. The values which can be set are from 0 to 4 and correspond to the pen styles listed in the Stroke attributes properties.

This property if not fully supported in Windows CE (accepts only the 0 and 1 values).

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        Dim iRet As Integer
        PenStyle = 1
        iRet = PenStyle
        MsgBox "PenStyle = " & CStr(iRet), vbOkOnly, "Test PenStyle"
        PenStyle = 0
        iRet = PenStyle
        MsgBox "PenStyle = " & CStr(iRet), vbOkOnly, "Test PenStyle"
        PenStyle = 4
        iRet = PenStyle
        MsgBox "PenStyle = " & CStr(iRet), vbOkOnly, "Test PenStyle"
End Sub
```

PenVisible, DrawCmdTarget Property

Syntax Visible = Boolean

Description This property sets or returns the logic condition of the Pen's visibility. Accepts a Boolean type parameter.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim bRet As Boolean
        Visible = False
        bRet = Visible
        MsgBox "Visible = " & CStr(bRet), vbOkOnly, "Test Visible"
        Visible = True
        bRet = Visible
        MsgBox "Visible = " & CStr(bRet), vbOkOnly, "Test Visible"
End Sub
```

PenWidth, DrawCmdTarget Property

Syntax PenWidth = Integer

Description Sets or returns the object's border line size in pixels.

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        Dim iRet As Integer
        PenWidth = 1
        iRet = PenWidth
        MsgBox "PenWidth = " & CStr(iRet), vbOkOnly, "Test PenWidth"
        PenWidth = 5
        iRet = PenWidth
        MsgBox "PenWidth = " & CStr(iRet), vbOkOnly, "Test PenWidth"
        PenWidth = 10
        iRet = PenWidth
        MsgBox "PenWidth = " & CStr(iRet), vbOkOnly, "Test PenWidth"
End Sub
```

Rotation, DrawCmdTarget Property

Syntax Rotation = Integer

Description This property returns or sets the rotation value assigned to value in degrees from 0 to 360. This can also be set in the same object's properties. Accepts an Integer type parameter.

This property is not supported in Windows CE.(If set, alwyas returns zero)

Parameter	Description

None	None

Result Integer

Example:

```
Public Sub Click()
        Dim iRet As Integer
        Rotation = 30
        iRet = Rotation
        MsgBox "Rotation = " & CStr(iRet), vbOkOnly, "Test Rotation"
        Rotation = 45
        iRet = Rotation
        MsgBox "Rotation = " & CStr(iRet), vbOkOnly, "Test Rotation"
        Rotation = 60
        iRet = Rotation
        MsgBox "Rotation = " & CStr(iRet), vbOkOnly, "Test Rotation"
End Sub
```

Shadow, DrawCmdTarget Property

Syntax Shadow = _Boolean

Description This property enables or disables the object's shadow.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Shadow = Not Shadow
        sRet = Shadow
        MsgBox "Shadow = " & sRet, vbOkOnly, GetProjectTitle
End Sub
```

ShadowColor, DrawCmdTarget Property

Syntax ShadowColor = _Long

Description This property sets or returns the color of the Object's shadow. Accepts a Long parameter containing the code (R,G,B, in each byte) for the shadow color. It may be more helpful to use the Movicon RGB function.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        ShadowColor =RGB(255,255,255)
        sRet = ShadowColor
        MsgBox "ShadowColor = " & sRet, vbOkOnly, GetProjectTitle 'Return-> 16777215
End Sub
```

ShadowXOffset, DrawCmdTarget Property

Syntax ShadowXOffset = _Integer

Description This property sets or returns the horizontal offset of the shadow referred to the drawing object where it is displayed. Positive values indicate one move to the shadow's right while negative values indicate one move to the left. The shadow is displayed in the set offset only when it is enabled.

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        Shadow = True
        ShadowXOffset =10
        lRet = ShadowXOffset
        MsgBox "ShadowXOffset = " & CStr(lRet), vbOkOnly, "Test ShadowXOffset "
End Sub
```

ShadowYOffset, DrawCmdTarget Property

Syntax ShadowYOffset = _Integer

Description This property sets or returns the vertical offset of the shadow referred to the drawing object where it is displayed. Positive values indicate one move to the shadow's right while negative values indicate one move to the left. The shadow is displayed in the set offset only when it is enabled.

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        Shadow = True
        ShadowYOffset =10
        lRet = ShadowYOffset
        MsgBox "ShadowYOffset = " & CStr(lRet), vbOkOnly, "Test ShadowYOffset "
End Sub
```

ShowFocusRectangle, DrawCmdTarget Property

Syntax ShowFocusRectangle = _Boolean

Description This property sets or returns the object's focus feature. The focus is represented by a dotted frame around the outside of the drawing which appears when the drawing is clicked on with the mouse or managed from the keyboard.
It is enabled to display around the drawing with the true value.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim bRet As Boolean
        ShowFocusRectangle = Not ShowFocusRectangle
        bRet = ShowFocusRectangle
        MsgBox "ShowFocusRectangle = " & CStr(bRet), vbOkOnly, "Test ShowFocusRectangle "
End Sub
```

ShowHiliteRectangle, DrawCmdTarget Property

Syntax ShowHiliteRectangle = _Boolean

Description Sets or returns the Highlight property which highlights the drawing object when the mouse passes over or is on it. The highlight is done by raising the border around the drawing which appears when the drawing becomes mouse or keyboard sensitive.
The highlight is enabled with the true value.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim bRet As Boolean
        ShowHiliteRectangle = Not ShowHiliteRectangle
        bRet = ShowHiliteRectangle
        MsgBox "ShowHiliteRectangle = " & CStr(bRet), vbOkOnly, "Test ShowHiliteRectangle "
End Sub
```

StatusVariable, DrawCmdTarget Property

Syntax StatusVariable = _String

Description This property sets or returns the name of the status variable associated to the referenced object. The symbol will assume a different graphic state according to the variable's status quality so that an immediate visual understanding of the variable's status can be obtained.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim objRect As DrawCmdTarget
        Dim sVarName As String

        GetVariableNameFromList(sVarName)
        Set objRect = GetSynopticObject.GetSubObject("objRect")
        objRect.StatusVariable = sVarName

        Set objRect = Nothing
End Sub
```

SynapseValue, DrawCmdTarget Property

Syntax SynapseValue(_lpszSynapseName)

Description This property sets or returns the value of a synapse set in the drawing object by identifying the synapse's name through the lpszSynapseName parameter.
The set or returned value is variant type and allows synapse compatibility with all data types set with the basic script language.

Parameter	Description
lpszSynapseName As String	Synapse's name.

Result Variant

Example:

```
Public Sub OnFireExecution()
        Dim Result As Variant
        ' Read the old value (Legge il valore vecchio)
        Result = GetVariableValue("Temperature")
        ' Set value (Imposta il valore)
        SynapseValue("OUT") = Result
End Sub
```

SynapsisVisible, DrawCmdTarget Property

Syntax SynapsisVisible = Boolean

Description This property sets or returns the visibility status of the synapses inserted into the drawing.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        SynapsisVisible = Not SynapsisVisible
        MsgBox "SynapsisVisible = " & CStr(SynapsisVisible ), vbOkOnly, "Test SynapsisVisible "
End Sub
```

Title, DrawCmdTarget Property

Syntax Title = String

Description This property returns or sets the title assigned to the control. This can also be set by means of using the control's general properties. The title will appear as static text in the object.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sRet As String
        Title = "MyTitle"
        sRet = Title
```

```
        MsgBox "Title = " & sRet, vbOkOnly, "Test Title"
End Sub
```

ToolTip, DrawCmdTarget Property

Syntax ToolTip = _String

Description This property sets or returns the string of the ToolTip relating to the object.

This property is not supported in Windows CE. (If set, returns an empty string)

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sRet As String
        ToolTip = "OggetoX"
        sRet = ToolTip
        MsgBox "ToolTip = " & sRet, vbOkOnly, "Test ToolTip"
End Sub
```

Transparency, DrawCmdTarget Property

Syntax Transparency = _Integer

Description This property sets or returns the object' transparency level. The values start from 0 to 255 (0 = completely invisible, 255 = completely visible).

This property is not supported in Windows CE. (If set, always returns zero).

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        Transparency = 255
End Sub
```

VariableBackColor, DrawCmdTarget Property

Syntax VariableBackColor = String

Description This property returns or sets the name of variable associated to the object's back color.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sRet As String
        VariableBackColor = "VAR0001"
        sRet = VariableBackColor
        MsgBox "VariableBackColor = " & sRet, vbOkOnly, "Test VariableBackColor"
End Sub
```

VariableBitmapList, DrawCmdTarget Property

Syntax VariableBitmapList = String

Description This property returns or sets the name of the reference variable for the object's bitmap image property.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sRet As String
        VariableBitmapList = "VAR0001"
        sRet = VariableBitmapList
        MsgBox "VariableBitmapList = " & sRet, vbOkOnly, GetProjectTitle
End Sub
```

VariableComposedMovement, DrawCmdTarget Property

Syntax VariableComposedMovement = _String

Description This property returns or sets the name of the reference variable for the object's composed movement property.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        MsgBox "VariableComposedMovement = " & VariableComposedMovement(), vbOkOnly,
        GetProjectTitle
End Sub
```

VariableEdgeColor, DrawCmdTarget Property

Syntax VariableEdgeColor = String

Description This property returns or sets the name of the variable associated to the object's edge and text color.

Parameter	Description
lpar As String	Variable's name.

Result String

Example:

```
Public Sub Click()
        Dim sRet As String
        VariableEdgeColor = "VAR0001"
        sRet = VariableEdgeColor
        MsgBox "VariableEdgeColor = " & sRet, vbOkOnly, GetProjectTitle
End Sub
```

VariableEndingX, DrawCmdTarget Property

Syntax VariableEndingX = String

Description This property returns or sets the name of the reference variable for the object's Ending X property.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sRet As String
        VariableEndingX = "VAR0001"
        sRet = VariableEndingX
        MsgBox "VariableEndingX = " & sRet, vbOkOnly, GetProjectTitle
End Sub
```

VariableEndingY, DrawCmdTarget Property

Syntax VariableEndingY = String

Description This property returns or sets the name of the reference variable for the object's Ending Y property.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sRet As String
        VariableEndingY = "VAR0001"
        sRet = VariableEndingY
        MsgBox "VariableEndingY = " & sRet, vbOkOnly, GetProjectTitle
End Sub
```

VariableFillColor, DrawCmdTarget Property

Syntax VariableFillColor = String

Description This property returns or sets the name of the reference variable for the object's color animation.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sRet As String
        VariableFillColor = "VAR0001"
        sRet = VariableFillColor
```

```
        MsgBox "VariableFillColor = " & sRet, vbOkOnly, GetProjectTitle
End Sub
```

VariableFilling, DrawCmdTarget Property

Syntax VariableFilling = String

Description This property returns or sets the name of the reference variable for the object's filling animation.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sRet As String
        VariableFilling = "VAR0001"
        sRet = VariableFilling
        MsgBox "VariableFilling = " & sRet, vbOkOnly, GetProjectTitle
End Sub
```

VariableLinkedText, DrawCmdTarget Property

Syntax VariableLinkedText = String

Description This property returns or sets the name of the variable linked to the object's text property.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sRet As String
        VariableLinkedText = "VAR0001"
        sRet = VariableLinkedText
        MsgBox "VariableLinkedText = " & sRet, vbOkOnly, GetProjectTitle
End Sub
```

VariableMoveX, DrawCmdTarget Property

Syntax VariableMoveX = String

Description This property returns or sets the name of the reference variable for the object's horizontal (X) move property.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sRet As String
        VariableMoveX = "VAR0001"
        sRet = VariableMoveX
        MsgBox "VariableMoveX = " & sRet, vbOkOnly, GetProjectTitle
End Sub
```

VariableMoveY, DrawCmdTarget Property

Syntax VariableMoveY = String

Description This property returns or sets the name of the reference variable for the object's vertical (V) move property.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sRet As String
        VariableMoveY = "VAR0001"
        sRet = VariableMoveY
        MsgBox "VariableMoveY = " & sRet, vbOkOnly, GetProjectTitle
End Sub
```

VariableRotation, DrawCmdTarget Property

Syntax VariableRotation = String

Description This property returns or sets the name of the reference variable for the object's rotation property.

This property is not supported in Windows CE.(If set, always returns an empty string)

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sRet As String
        VariableRotation = "VAR0001"
        sRet = VariableRotation
        MsgBox "VariableRotation = " & sRet, vbOkOnly, GetProjectTitle
End Sub
```

VariableScaling, DrawCmdTarget Property

Syntax VariableScaling = String

Description This property returns or sets the name of the reference variable for the object's Scaling property.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sRet As String
        VariableScaling = "VAR0001"
        sRet = VariableScaling
        MsgBox "VariableScaling = " & sRet, vbOkOnly, GetProjectTitle
End Sub
```

VariableStartingX, DrawCmdTarget Property

Syntax VariableStartingX = String

Description This property returns or sets the name of the reference variable for the object's Start X Point property.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sRet As String
        VariableStartingX = "VAR0001"
        sRet = VariableStartingX
        MsgBox "VariableStartingX = " & sRet, vbOkOnly, GetProjectTitle
End Sub
```

VariableStartingY, DrawCmdTarget Property

Syntax VariableStartingY = String

Description This property returns or sets the name of the reference variable for the object's Start Y Point property.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sRet As String
        VariableStartingY = "VAR0001"
        sRet = VariableStartingY
        MsgBox "VariableStartingY  = " & sRet, vbOkOnly, GetProjectTitle
End Sub
```

VariableVisible, DrawCmdTarget Property

Syntax VariableVisible = String

Description This property returns or sets the name of the variable associated to the object's Visible property.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sRet As String
        VariableVisible = "VAR0001"
```

```
        sRet = VariableVisible
        MsgBox "VariableVisible = " & sRet, vbOkOnly, "Test VariableVisible "
End Sub
```

VariableXRotationCenter, DrawCmdTarget Property

Syntax VariableXRotationCenter = String

Description This property returns or sets the name of the reference variable for the X Point of the drawing's Center Rotation property.

This property is not supported in Windows CE.(If set, always returns an empty string)

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sRet As String
        VariableXRotationCenter = "VAR0001"
        sRet = VariableXRotationCenter
        MsgBox "VariableXRotationCenter = " & sRet, vbOkOnly, GetProjectTitle
End Sub
```

VariableYRotationCenter, DrawCmdTarget Property

Syntax VariableYRotationCenter = String

Description This property returns or sets the name of the reference variable for the Y Point of the drawing's Center Rotation property.

This property is not supported in Windows CE.(If set, always returns an empty string)

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sRet As String
        VariableYRotationCenter = "VAR0001"
        sRet = VariableYRotationCenter
        MsgBox "VariableYRotationCenter = " & sRet, vbOkOnly, GetProjectTitle
End Sub
```

Visible, DrawCmdTarget Property

Syntax Visible = Boolean

Description Sets or returns the object's visible condition.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Visible = Not Visible
        MsgBox "Visible = " & CStr(Visible), vbOkOnly, "Test Visible"
End Sub
```

Width, DrawCmdTarget Property

Syntax Width = _Long

Description This property returns or sets the value assigned to the size of the object's width.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim lRet As Long
        Width = Width + 10
        lRet = Width
        MsgBox "Width = " & CStr(lRet), vbOkOnly, GetProjectTitle
End Sub
```

Xpos, DrawCmdTarget Property

Syntax Xpos = Long

Description This property expresses the X coordinate of the component's furthest left corner edge. The value is expressed in pixels and relates to the Screen window's 0 point (the top left corner edge) which contains the component. The value of this property will be automatically modified each time the component is moved in the Screen and vice-versa by modifying this value the component will graphically change its position.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim lRet As Long
        Xpos = Xpos + 1
        lRet = Xpos
        MsgBox "Xpos = " & CStr(lRet), vbOkOnly, "Test Xpos"
End Sub
```

XRotationCenter, DrawCmdTarget Property

Syntax XRotationCenter = _Integer

Description This property sets or returns the X coordinate of the object's rotation center.

This property is not supported in Windows CE. (If set, always returns zero)

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        Dim iRet As Integer
        XRotationCenter = 10
        iRet = XRotationCenter
        MsgBox "XRotationCenter = " & CStr(iRet), vbOkOnly, "Test XRotationCenter"
End Sub
```

Ypos, DrawCmdTarget Property

Syntax Ypos = Long

Description This property expresses the Y coordinate of the component's highest corner edge. The value is expressed in pixels and relates to the Screen window's 0 point (the top left corner edge) which contains the component. The value of this property will be automatically modified each time the component is moved in the Screen and vice-versa by modifying this value the component will graphically change its position.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim lRet As Long
        Ypos = Ypos + 1
        lRet = Ypos
        MsgBox "Ypos = " & CStr(lRet), vbOkOnly, "Test Ypos"
End Sub
```

YRotationCenter, DrawCmdTarget Property

Syntax YRotationCenter = _Integer

Description This property sets or returns the Y coordinate of the object's rotation center.

This property is not supported in Windows CE. (If set, always returns zero)

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        Dim iRet As Integer
        YRotationCenter = 10
        iRet = YRotationCenter
        MsgBox "YRotationCenter = " & CStr(iRet), vbOkOnly, "Test YRotationCenter "
End Sub
```

30. EventCmdTarget

GetXMLSettings, EventCmdTarget Function

Syntax GetXMLSettings()

Description This function returns a string with the contents of the project's XML file relating to the referred event.

Parameter	Description
None	None

Result String

Example:

```
Public Sub AlarmLoading()
        Dim EventObj As EventCmdTarget
        Set EventObj = GetEvent("Event1")
        If Not EventObj Is Nothing Then
                MsgBox EventObj.GetXMLSettings ,vbOkOnly,""
        End If
        Set EventObj = Nothing
End Sub
```

CommandList, EventCmdTarget Property

Syntax CommandList = _String

Description This property returns the project's XML string containing the definitions of the commands associated to the reference event.

Parameter	Description
None	None

Result String

Example:

```
Public Sub AlarmLoading()
        Dim EventObj As EventCmdTarget
        Set EventObj = GetEvent("Event1")
        If Not EventObj Is Nothing Then
                MsgBox EventObj.CommandList,vbOkOnly,""
        End If
        Set EventObj = Nothing
End Sub
```

Condition, EventCmdTarget Property

Syntax Condition = _Integer

Description This property returns or sets the condition to verify the reference event.

The possible configurations are:

enum_evc_changed
enum_evc_equal
enum_evc_major
enum_evc_minor

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub AlarmLoading()
        Dim EventObj As EventCmdTarget
        Set EventObj = GetEvent("Event1")
        If Not EventObj Is Nothing Then
                EventObj.Condition = enum_evc_equal
        End If
        Set EventObj = Nothing
End Sub
```

Enable, EventCmdTarget Property

Syntax Enable = _Boolean

Description This property enables or disables the reference event.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub AlarmLoading()
        Dim EventObj As EventCmdTarget
        Set EventObj = GetEvent("Event1")
        If Not EventObj Is Nothing Then
                EventObj.Enable = True
        End If
        Set EventObj = Nothing
End Sub
```

EnableVariable, EventCmdTarget Property

Syntax EnableVariable = _String

Description This property sets or returns the name of the enable variable for the reference event.
When the variable's value is equal to "zero", the "Command" of the Event Object will not be executed during Runtime. When its value is different from "zero", the "Command" of the Event Object will be executed according to the set conditions.

*This function is managed by Movicon only when the "Enable" property has been activated. Otherwise the Event Object will not execute any commands independently of the **"Enable Variable"** status.*

Parameter	Description
None	None

Result String

Example:

```
Public Sub AlarmLoading()
        Dim EventObj As EventCmdTarget
        Set EventObj = GetEvent("Event1")
        If Not EventObj Is Nothing Then
                EventObj.EnableVariable = "VAR00004"
        End If
        Set EventObj = Nothing
End Sub
```

Name, EventCmdTarget Property

Syntax Name = _String

Description This function returns a string with the name of the reference event object.

Parameter	Description
None	None

Result String

Example:

```
Public Sub AlarmLoading()
        Dim EventObj As EventCmdTarget
        Set EventObj = GetEvent("Event1")
        If Not EventObj Is Nothing Then
                MsgBox "Event Name is " & EventObj.Name,vbOkOnly,""
        End If
        Set EventObj = Nothing
End Sub
```

Value, EventCmdTarget Property

Syntax Value = _Double

Description This property returns or sets the threshold value to which the "Condition" property refers. This setting has no meaning when the "Change" has been selected as "Condition".

Parameter	Description
None	None

Result Double

Example:

```
Public Sub AlarmLoading()
        Dim EventObj As EventCmdTarget
        Set EventObj = GetEvent("Event1")
        If Not EventObj Is Nothing Then
                EventObj.Value = 12
        End If
        Set EventObj = Nothing
End Sub
```

Variable, EventCmdTarget Property

Syntax Variable = _String

Description This property sets or returns the name of the variable to be monitored. When its value changes, if required by the "Condition" property settings, the associated "Command" will be executed.

Parameter	Description
None	None

Result String

Example:

```
Public Sub AlarmLoading()
        Dim EventObj As EventCmdTarget
        Set EventObj = GetEvent("Event1")
        If Not EventObj Is Nothing Then
                EventObj.Variable = "VAR00005"
        End If
        Set EventObj = Nothing
End Sub
```

31. GaugeCmdTarget

LoadExtSettings, GaugeCmdTarget Function

Syntax LoadExtSettings

Description This function permits the object's relating external file settings to be loaded. This file can be specified in design mode in the "External File settings" property or in the "ExtSettingsFile" interface properties. The extension provided for this file is ".SXML".

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
Dim objSymbol As GaugeCmdTarget
Set objSymbol = GetSynopticObject.GetSubObject("TestObject").GetObjectInterface
If objSymbol Is Nothing Then Exit Sub
objSymbol.ExtSettingsFile = "test.sxml"
objSymbol.LoadExtSettings
Set objSymbol = Nothing
End Sub
```

SaveExtSettings, GaugeCmdTarget Function

Syntax SaveExtSettings

Description This function permits the objects settings to be save in the relating external settings file. This file can be specified when in design mode in the "Ext. Settings File" property, or using the property from the "ExtSettingsFile" interface. The extension provided for this file is ".SXML".

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
Dim objSymbol As GaugeCmdTarget
Set objSymbol = GetSynopticObject.GetSubObject("TestObject").GetObjectInterface
If objSymbol Is Nothing Then Exit Sub
objSymbol.ExtSettingsFile = "test.sxml"
```

```
objSymbol.SaveExtSettings
Set objSymbol = Nothing
End Sub
```

BarBackColor, GaugeCmdTarget Property

Syntax BarBackColor = _Long

Description This property sets or returns the color to be associated to the back color of the Gauge's bar.

Parameter	Description
None	None

Result Long

Example:

```
Option Explicit
Public Sub Click()
	Dim MyGauge As GaugeCmdTarget
	Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
	If Not MyGauge Is Nothing Then
		MyGauge.BarBackColor = RGB(234,176,89)
		Set MyGauge = Nothing
	End If
End Sub
```

BarBias, GaugeCmdTarget Property

Syntax BarBias = _Boolean

Description This property allows the "BarBias", associated to the variable in the Gauge window, to be displayed or hidden.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
	Dim MyGauge As GaugeCmdTarget
	Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
	If Not MyGauge Is Nothing Then
		MyGauge.BarBias = Not MyGauge.BarBias
		Set MyGauge = Nothing
	End If
End Sub
```

BarBorder, GaugeCmdTarget Property

Syntax BarBorder = _Integer

Description This property sets or returns the chart border type of the Gauge bar. The values can be from 0 to 7. This property has meaning only when the Gauge is vertical or horizontal type.

The values refer to:

0 none
1 normal
2 medium
3 thick
4 bump
5 etched
6 raised
7 sunken

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim i As Integer
Public Sub Click()
        Dim MyGauge As GaugeCmdTarget
        Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
        If Not MyGauge Is Nothing Then
                MyGauge.BarVisible = True
                MyGauge.BarBorder = i
                i = i + 1
                If i = 8 Then
                i = 0
                End If
                Set MyGauge = Nothing
        End If
End Sub
Public Sub SymbolLoading()
        i = 0
End Sub
```

BarFillColor, GaugeCmdTarget Property

Syntax BarFillColor = _Long

Description This property sets or returns the color to be associated to filling the Gauge bar according to the value of the associated variable.

Parameter	Description

None	None

Result Long

Example:

```
Option Explicit
Public Sub Click()
	Dim MyGauge As GaugeCmdTarget
	Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
	If Not MyGauge Is Nothing Then
		MyGauge.BarFillColor = RGB(234,176,89)
		Set MyGauge = Nothing
	End If
End Sub
```

BarVisible, GaugeCmdTarget Property

Syntax BarVisible = _Boolean

Description This property allows the back "Bar" and its filling to be displayed or hidden in the Gauge window.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
	Dim MyGauge As GaugeCmdTarget
	Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
	If Not MyGauge Is Nothing Then
		MyGauge.BarVisible = Not MyGauge.BarVisible
		Set MyGauge = Nothing
	End If
End Sub
```

CenterPos, GaugeCmdTarget Property

Syntax CenterPos = _Integer

Description This property sets or returns the Gauge's needle center position and, as a consequence, all the semicircle of elements as well.
The possible values are:

0 top left
1 top centre
2 top right
3 centre left
4 centre

5 centre right
6 bottom left
7 centre bottom
8 bottom right

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim i As Integer
Public Sub Click()
	Dim MyGauge As GaugeCmdTarget
	Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
	If Not MyGauge Is Nothing Then
		MyGauge.BarVisible=True
		MyGauge.CenterPos = i
		i = i + 1
		If i = 9 Then
		i = 0
		End If
		Set MyGauge = Nothing
	End If
End Sub
Public Sub SymbolLoading()
	i = 0
End Sub
```

ColorWarningZone, GaugeCmdTarget Property

Syntax ColorWarningZone(_nZone) = _Long

Description This property sets or returns the color to be associated to the Warning Zone of the Gauge referenced by the index. There are 5 zones and relating indexes are from zero to 4.

Parameter	Description
nZone As Integer	Index of the reference zone.

Result Long

Example:

```
Option Explicit
Public Sub Click()
	Dim MyGauge As GaugeCmdTarget
	Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
	If Not MyGauge Is Nothing Then
		MyGauge.ColorWarningZone(0) = RGB(234,176,89)
		Set MyGauge = Nothing
	End If
```

```
End Sub
```

EnableWarningZone, GaugeCmdTarget Property

Syntax EnableWarningZone(_nZone) = _Boolean

Description This property sets or returns the visibility of the Warning Zone of the Gauge referenced by the index. There are 5 zones and the relating indexes start from zero to 4.

Parameter	Description
nZone As Integer	Index of the reference zone

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
	Dim MyGauge As GaugeCmdTarget
	Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
	If Not MyGauge Is Nothing Then
		MyGauge.EnableWarningZone(0) = True
		Set MyGauge = Nothing
	End If
End Sub
```

EndAngle, GaugeCmdTarget Property

Syntax EndAngle = _Integer

Description This property sets or returns the value of the angle relating to the end of the circle arc which represents the elements of the gauge. This property only has meaning when the gauge it circular type.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim i As Integer
Public Sub Click()
	Dim MyGauge As GaugeCmdTarget
	Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
	If Not MyGauge Is Nothing Then
		MyGauge.BarVisible = True
		MyGauge.EndAngle = i
```

```
                i = i + 1
                If i = 360 Then
                i = 0
                End If
                Set MyGauge = Nothing
        End If
End Sub
Public Sub SymbolLoading()
        i = 0
End Sub
```

EndWarningZone, GaugeCmdTarget Property

Syntax EndWarningZone(_nZone) = _Integer

Description This property sets or returns the End value of the Warning Zone or the Gauge reference by the index. There are 5 zones and the relating indexes start from zero to 4.

Parameter	Description
nZone As Integer	Index of the reference zone

Result Integer

Example:

```
Option Explicit
Public Sub Click()
        Dim MyGauge As GaugeCmdTarget
        Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
        If Not MyGauge Is Nothing Then
                MyGauge.EndWarningZone(0) = 50
                Set MyGauge = Nothing
        End If
End Sub
```

ExtSettingsFile, GaugeCmdTarget Property

Syntax ExtSettingsFile = _String

Description This property sets or returns the external configuration file for the referenced object. the file can be alsow specified in design mode in the object's "Configuration File" property. The extension provided for this file is ".SXML".

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
Dim objSymbol As GaugeCmdTarget
Set objSymbol = GetSynopticObject.GetSubObject("TestObject").GetObjectInterface
If objSymbol Is Nothing Then Exit Sub
objSymbol.ExtSettingsFile = "test.sxml"
objSymbol.SaveExtSettings
Set objSymbol= Nothing
End Sub
```

FontHeightScale, GaugeCmdTarget Property

Syntax FontHeightScale = _Integer

Description This property sets or returns the size of the font used in the Gauge's scale.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
	Dim MyGauge As GaugeCmdTarget
	Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
	If Not MyGauge Is Nothing Then
		MyGauge.FontNameScale = "Arial"
		MyGauge.FontHeightScale = 20
		Set MyGauge = Nothing
	End If
End Sub
```

FontNameScale, GaugeCmdTarget Property

Syntax FontNameScale = _String

Description This property sets or returns the font used in the Gauge's scale.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
	Dim MyGauge As GaugeCmdTarget
```

```
        Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
        If Not MyGauge Is Nothing Then
            MyGauge.FontNameScale = "Arial"
            MyGauge.FontHeightScale = 20
            Set MyGauge = Nothing
        End If
End Sub
```

FormatVariable, GaugeCmdTarget Property

Syntax FormatVariable = _String

Description This property returns or sets the name of the variable whose value determines the display format of the Gauge's scale.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim objGauge As GaugeCmdTarget
        Set objGauge = GetSynopticObject.GetSubObject("Gauge1").GetObjectInterface
        MsgBox  "Gauge1 Format Variable = " & objGauge.FormatVariable,
        vbInformation,GetProjectTitle
        Set objGauge = Nothing
End Sub
```

GaugeMaxVariable, GaugeCmdTarget Property

Syntax GaugeMaxVariable = _String

Description This property sets or returns the name of the variable associated to the maximum threshold represented in the Gauge.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim MyGauge As GaugeCmdTarget
        Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
        If Not MyGauge Is Nothing Then
```

```
            MyGauge.GaugeMaxVariable = "VAR00001"
            VAR00001 = 90
            Set MyGauge = Nothing
        End If
End Sub
```

GaugeMinVariable, GaugeCmdTarget Property

Syntax GaugeMinVariable = _String

Description This property sets or returns the name of variable associated to the minimum threshold represented in the Gauge.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim MyGauge As GaugeCmdTarget
        Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
        If Not MyGauge Is Nothing Then
            MyGauge.GaugeMinVariable = "VAR00002"
            VAR00002 = 10
            Set MyGauge = Nothing
        End If
End Sub
```

GaugeType, GaugeCmdTarget Property

Syntax GaugeType = _Integer

Description This property sets or returns the Gauge type to be represented.

The values are:

0 vertical
1 horizontal
2 circular

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim i As Integer
Public Sub Click()
        Dim MyGauge As GaugeCmdTarget
        Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
        If Not MyGauge Is Nothing Then
                MyGauge.GaugeType=  i
                i = i + 1
                If i = 3 Then
                        i = 0
                End If
                Set MyGauge = Nothing
        End If
End Sub
Public Sub SymbolLoading()
        i = 0
End Sub
```

GaugeVariable, GaugeCmdTarget Property

Syntax GaugeVariable = _String

Description This property sets or returns the name of the variable represented in the Gauge.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim MyGauge As GaugeCmdTarget
        Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
        If Not MyGauge Is Nothing Then
                MyGauge.GaugeVariable = "VAR00004"
                Set MyGauge = Nothing
        End If
End Sub
```

GaugeWarningZoneEndVariable, GaugeCmdTarget Property

Syntax GaugeWarningZoneEndVariable(_nZone) = _String

Description This property sets or returns the name of the variable that manages the end value of the Gauge's Alarm Zone. The "nZone" parameter indicated the reference zone. There are 5 zones and related indexes start from 0 to 4.

Parameter	Description

nZone As Integer	This value indicates the referenced alarm zone number.

Result String

Example:

```
Option Explicit
Public Sub Click()
	Dim MyGauge As GaugeCmdTarget
	Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
	If Not MyGauge Is Nothing Then
		MyGauge.GaugeWarningZoneEndVariable(0) = "VarStartZone1"
		Set MyGauge = Nothing
	End If
End Sub
```

GaugeWarningZoneStartVariable, GaugeCmdTarget Property

Syntax GaugeWarningZoneStartVariable(_nZone) = _String

Description This property sets or returns the name of the variable that manages the start value of the Gauge's Alarm Zone. The "nZone" parameter indicated the reference zone.

Parameter	Description
nZone As Integer	This value indicates the referenced alarm zone number.

Result String

Example:

```
Option Explicit
Public Sub Click()
	Dim MyGauge As GaugeCmdTarget
	Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
	If Not MyGauge Is Nothing Then
		MyGauge.GaugeWarningZoneStartVariable(0) = "VarStartZone1"
		Set MyGauge = Nothing
	End If
End Sub
```

GeneralGap, GaugeCmdTarget Property

Syntax GeneralGap = _Integer

Description This property sets or returns the value of the gap between the various graphical parts (window border/ warning zone/bar) of the Gauge object.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim i As Integer
Public Sub Click()
        Dim MyGauge As GaugeCmdTarget
        Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
        If Not MyGauge Is Nothing Then
                MyGauge.GeneralGap =  i
                i = i + 1
                If i = 5 Then
                        i = 0
                End If
                Set MyGauge = Nothing
        End If
End Sub
Public Sub SymbolLoading()
        i = 0
End Sub
```

InvertScale, GaugeCmdTarget Property

Syntax InvertScale = _Boolean

Description This property enables or disables the Gauge scale's value inversion.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim i As Integer
Public Sub Click()
        Dim MyGauge As GaugeCmdTarget
        Set MyGauge =
        GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
        If Not MyGauge Is Nothing Then
                MyGauge.InvertScale = Not MyGauge.InvertScale
                Set MyGauge = Nothing
        End If
End Sub
Public Sub SymbolLoading()
        i = 0
End Sub
```

KnobBorder, GaugeCmdTarget Property

Syntax KnobBorder = _Integer

Description This property sets or returns the knob border type. The possible values are from 0 to 7. This property has meaning only when the Gauge type is vertical or horizontal.

0 none
1 normal
2 medium
3 thick
4 bump
5 etched
6 raised
7 sunken

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim i As Integer
Public Sub Click()
        Dim MyGauge As GaugeCmdTarget
        Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
        If Not MyGauge Is Nothing Then
                MyGauge.BarVisible = True
                MyGauge.KnobBorder = i
                i = i + 1
                If i = 8 Then
                i = 0
                End If
                Set MyGauge = Nothing
        End If
End Sub
Public Sub SymbolLoading()
        i = 0
End Sub
```

KnobColor, GaugeCmdTarget Property

Syntax KnobColor = _Long

Description This property sets or returns the color of the Guage's knob.

Parameter	Description
None	None

Result Long

Example:

```
Option Explicit
Dim i As Integer
Public Sub Click()
        Dim MyGauge As GaugeCmdTarget
        Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
        If Not MyGauge Is Nothing Then
                MyGauge.BarVisible = True
                MyGauge.KnobColor = RGB(125,7,90)
                Set MyGauge = Nothing
        End If
End Sub
Public Sub SymbolLoading()
        i = 0
End Sub
```

LabelEvery, GaugeCmdTarget Property

Syntax LabelEvery = _Integer

Description This property sets or returns for every amount of major divisions of the Scale a numeric label is to be displayed to identify the value in that position.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim i As Integer
Public Sub Click()
        Dim MyGauge As GaugeCmdTarget
        Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
        If Not MyGauge Is Nothing Then
                MyGauge.BarVisible = True
                MyGauge.LabelEvery = 5
                Set MyGauge = Nothing
        End If
End Sub
Public Sub SymbolLoading()
        i = 0
End Sub
```

MaxValue, GaugeCmdTarget Property

Syntax MaxValue = _Integer

Description This property sets or returns the maximum value to be represented in the Gauge. When the Var. Max Limit property is set with nothing, this value will be ignored and the value of this variable will be considered as max.

scale settings.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim i As Integer
Public Sub Click()
	Dim MyGauge As GaugeCmdTarget
	Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
	If Not MyGauge Is Nothing Then
		MyGauge.BarVisible = True
		MyGauge.MaxValue = 120
		Set MyGauge = Nothing
	End If
End Sub
Public Sub SymbolLoading()
	i = 0
End Sub
```

MinValue, GaugeCmdTarget Property

Syntax MinValue = _Integer

Description This property sets or returns the maximum value to be represented in the Gauge. When the MinVar. limit is nothing, this value will be ignored and the value of the same variable will be considered as the scale's beginning.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim i As Integer
Public Sub Click()
	Dim MyGauge As GaugeCmdTarget
	Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
	If Not MyGauge Is Nothing Then
		MyGauge.BarVisible = True
		MyGauge.MinValue = 10
		Set MyGauge = Nothing
	End If
End Sub
Public Sub SymbolLoading()
	i = 0
End Sub
```

NeedleBorder, GaugeCmdTarget Property

Syntax NeedleBorder = _Integer

Description This property sets or returns the Gauge's needle border type. The values are from 0 to 7. This property has significance only when the Gauge is circular type.

0 none
1 normal
2 medium
3 thick
4 bump
5 etched
6 raised
7 sunken

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim i As Integer
Public Sub Click()
	Dim MyGauge As GaugeCmdTarget
	Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
	If Not MyGauge Is Nothing Then
		MyGauge.BarVisible = True
		MyGauge.NeedleBorder = i
		i = i + 1
		If i = 8 Then
		i = 0
		End If
		Set MyGauge = Nothing
	End If
End Sub
Public Sub SymbolLoading()
	i = 0
End Sub
```

NeedleBorderColor, GaugeCmdTarget Property

Syntax NeedleBorderColor = _Long

Description This property sets or returns the color of the Gauge's needle border. This property has significance only when the Gauge is circular.

Parameter	Description

None	None

Result Long

Example:

```
Option Explicit
Dim i As Integer
Public Sub Click()
	Dim MyGauge As GaugeCmdTarget
	Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
	If Not MyGauge Is Nothing Then
		MyGauge.NeedleBorderColor = RGB(125,34,78)
		Set MyGauge = Nothing
	End If
End Sub
Public Sub SymbolLoading()
	i = 0
End Sub
```

NeedleColor, GaugeCmdTarget Property

Syntax NeedleColor = _Long

Description This property sets or returns the color of the Gauge's needle. This property has significance only when the gauge is circular.

Parameter	Description
None	None

Result Long

Example:

```
Option Explicit
Dim i As Integer
Public Sub Click()
	Dim MyGauge As GaugeCmdTarget
	Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
	If Not MyGauge Is Nothing Then
		MyGauge.NeedleColor = RGB(125,34,78)
		Set MyGauge = Nothing
	End If
End Sub
Public Sub SymbolLoading()
	i = 0
End Sub
```

NeedleLength, GaugeCmdTarget Property

Syntax NeedleLength = _Integer

Description This property sets or returns the Gauge needle's length. This property has significance only when the Gauge is circular.

The values are:

0 small
1 medium
2 large

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim i As Integer
Public Sub Click()
	Dim MyGauge As GaugeCmdTarget
	Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
	If Not MyGauge Is Nothing Then
		MyGauge.NeedleLength = RGB(125,34,78)
		i = i + 1
		If i = 3 Then
		i = 0
		End If
		Set MyGauge = Nothing
	End If
End Sub
Public Sub SymbolLoading()
	i = 0
End Sub
```

NeedleShadow, GaugeCmdTarget Property

Syntax NeedleShadow = _Boolean

Description This property enables or disables the Gauge's shadow. This property has significance only when the Gauge is circular.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim i As Integer
Public Sub Click()
	Dim MyGauge As GaugeCmdTarget
	Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
	If Not MyGauge Is Nothing Then
		MyGauge.NeedleShadow = Not MyGauge.NeedleShadow
```

```
                Set MyGauge = Nothing
        End If
End Sub
Public Sub SymbolLoading()
        i = 0
End Sub
```

NeedleShadowColor, GaugeCmdTarget Property

Syntax NeedleShadowColor = _Long

Description This property sets or returns the color of the Gauge needle's shadow. This property has meaning only when the Gauge is circular.

Parameter	Description
None	None

Result Long

Example:

```
Option Explicit
Dim i As Integer
Public Sub Click()
        Dim MyGauge As GaugeCmdTarget
        Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
        If Not MyGauge Is Nothing Then
                MyGauge.NeedleShadowColor = RGB(125,34,78)
                Set MyGauge = Nothing
        End If
End Sub
Public Sub SymbolLoading()
        i = 0
End Sub
```

NeedleVisible, GaugeCmdTarget Property

Syntax NeedleVisible = _Boolean

Description This property enables or disables the Gauge's needle and its shadow. This property has meaning only when the gauge is circular.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim i As Integer
Public Sub Click()
	Dim MyGauge As GaugeCmdTarget
	Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
	If Not MyGauge Is Nothing Then
		MyGauge.NeedleVisible = Not MyGauge.NeedleVisible
		Set MyGauge = Nothing
	End If
End Sub
Public Sub SymbolLoading()
	i = 0
End Sub
```

NeedleWidth, GaugeCmdTarget Property

Syntax NeedleWidth = _Integer

Description This property sets or returns the Gauge needle's width. This property has meaning only when the Gauge is circular.

The values may be:

0 small
1 medium
2 large

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim i As Integer
Public Sub Click()
	Dim MyGauge As GaugeCmdTarget
	Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
	If Not MyGauge Is Nothing Then
		MyGauge.NeedleWidth = RGB(125,34,78)
		i = i + 1
		If i = 3 Then
		i = 0
		End If
		Set MyGauge = Nothing
	End If
End Sub
Public Sub SymbolLoading()
	i = 0
End Sub
```

ScaleColor, GaugeCmdTarget Property

Syntax ScaleColor = _Long

Description This property sets or returns the color of the Gauge's scale.

Parameter	Description
None	None

Result Long

Example:

```
Option Explicit
Public Sub Click()
        Dim MyGauge As GaugeCmdTarget
        Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
        If Not MyGauge Is Nothing Then
                MyGauge.ScaleColor = RGB(125,34,78)
                Set MyGauge = Nothing
        End If
End Sub
```

ScaleFormat, GaugeCmdTarget Property

Syntax ScaleFormat = _String

Description This property sets or returns the format of the values represented on the Guage's scale.

The values are:

x Es: 1
xx Es: 01
xxx Es: 001
xxxx Es: 0001
xxxxx Es: 00001

x.x Es: 1.0
x.xx Es: 1.00
x.xxx Es: 1.000
x.xxxx Es: 1.0000
x.xxxxx Es: 1.00000

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim MyGauge As GaugeCmdTarget
        Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
        If Not MyGauge Is Nothing Then
                Debug.Print MyGauge.ScaleFormat
                Set MyGauge = Nothing
```

```
        End If
End Sub
```

ScaleMajorDiv, GaugeCmdTarget Property

Syntax ScaleMajorDiv = _Integer

Description This property sets or returns the number of major divisions to be displayed on the scale.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
        Dim MyGauge As GaugeCmdTarget
        Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
        If Not MyGauge Is Nothing Then
                MyGauge.ScaleMajorDiv = 6
                Set MyGauge = Nothing
        End If
End Sub
```

ScaleMinorDiv, GaugeCmdTarget Property

Syntax ScaleMinorDiv = _Integer

Description This property sets or returns the number of minor divisions to be displayed on the scale. The minor divisions are the ones between two major divisions.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
        Dim MyGauge As GaugeCmdTarget
        Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
        If Not MyGauge Is Nothing Then
                MyGauge.ScaleMinorDiv = 6
                Set MyGauge = Nothing
        End If
End Sub
```

ScaleRightBottom, GaugeCmdTarget Property

Syntax ScaleRightBottom = _Boolean

Description This property, when set to True boolean value, allow the Scale to be displayed on the Right hand side of the Gauge window when horizontal type or at the bottom if vertical type.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
        Dim MyGauge As GaugeCmdTarget
        Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
        If Not MyGauge Is Nothing Then
                MyGauge.ScaleRightBottom = Not MyGauge.ScaleRightBottom
                Set MyGauge = Nothing
        End If
End Sub
```

ScaleUnit, GaugeCmdTarget Property

Syntax ScaleUnit = _String

Description This property sets or returns the measure Unit on the Gauge's scale.

The possible values are:

x Es: 1
xx Es: 01
xxx Es: 001
xxxx Es: 0001
xxxxx Es: 00001

x.x Es: 1.0
x.xx Es: 1.00
x.xxx Es: 1.000
x.xxxx Es: 1.0000
x.xxxxx Es: 1.00000

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim MyGauge As GaugeCmdTarget
        Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
        If Not MyGauge Is Nothing Then
                MyGauge.ScaleUnit = "Hz"
                Set MyGauge = Nothing
        End If
End Sub
```

ScaleVisible, GaugeCmdTarget Property

Syntax ScaleVisible = _Boolean

Description This property enables or disables the visibility of the scale on the Gauge.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim i As Integer
Public Sub Click()
        Dim MyGauge As GaugeCmdTarget
        Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
        If Not MyGauge Is Nothing Then
                MyGauge.ScaleVisible = Not MyGauge.ScaleVisible
                Set MyGauge = Nothing
        End If
End Sub
Public Sub SymbolLoading()
        i = 0
End Sub
```

SliderBorder, GaugeCmdTarget Property

Syntax SliderBorder = _Integer

Description This property sets or returns the Gauge's border type. The values may be from 0 to 7. This property has meaning only when the Gauge is vertical or horizontal type.

0 none
1 normal
2 medium
3 thick
4 bump
5 etched
6 raised
7 sunken

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim i As Integer
Public Sub Click()
	Dim MyGauge As GaugeCmdTarget
	Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
	If Not MyGauge Is Nothing Then
		MyGauge.BarVisible = True
		MyGauge.SliderBorder = i
		i = i + 1
		If i = 8 Then
		i = 0
		End If
		Set MyGauge = Nothing
	End If
End Sub
Public Sub SymbolLoading()
	i = 0
End Sub
```

SliderColor, GaugeCmdTarget Property

Syntax SliderColor = _Long

Description This property sets or returns the color of the Gauge's Slider.

Parameter	Description
None	None

Result Long

Example:

```
Option Explicit
Dim i As Integer
Public Sub Click()
	Dim MyGauge As GaugeCmdTarget
	Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
	If Not MyGauge Is Nothing Then
		MyGauge.BarVisible = True
		MyGauge.SliderColor = RGB(125,7,90)
		Set MyGauge = Nothing
	End If
End Sub
Public Sub SymbolLoading()
	i = 0
End Sub
```

SliderVisible, GaugeCmdTarget Property

Syntax SliderVisible = _Boolean

Description This property displays or hides the Gauge's slider.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim i As Integer
Public Sub Click()
	Dim MyGauge As GaugeCmdTarget
	Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
	If Not MyGauge Is Nothing Then
		MyGauge.BarVisible = True
		MyGauge.SliderVisible = Not MyGauge.SliderVisible
		Set MyGauge = Nothing
	End If
End Sub
Public Sub SymbolLoading()
	i = 0
End Sub
```

StartAngle, GaugeCmdTarget Property

Syntax StartAngle = _Integer

Description This property sets or returns the value of the angle relating to the beginning of the circle's arc which represents the Gauge's elements. This property only has meaning when the gauge is circular type.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim i As Integer
Public Sub Click()
	Dim MyGauge As GaugeCmdTarget
	Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
	If Not MyGauge Is Nothing Then
		MyGauge.BarVisible = True
		MyGauge.StartAngle = i
```

```
                i = i + 1
                If i = 360 Then
                i = 0
                End If
                Set MyGauge = Nothing
        End If
End Sub
Public Sub SymbolLoading()
        i = 0
End Sub
```

StartWarningZone, GaugeCmdTarget Property

Syntax StartWarningZone(_nZone) = _Integer

Description This property sets or returns the start value of the Warning Zone of the Gauge referenced by the index. There are 5 zones and the related indexes are from zero to 4.

Parameter	Description
nZone As Integer	Index of the referenced zone.

Result Integer

Example:

```
Option Explicit
Public Sub Click()
        Dim MyGauge As GaugeCmdTarget
        Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
        If Not MyGauge Is Nothing Then
                MyGauge.StartWarningZone(0) = 10
                Set MyGauge = Nothing
        End If
End Sub
```

Title, GaugeCmdTarget Property

Syntax Title = _String

Description This property sets or returns the title displayed in the Gauge window.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
	Dim MyGauge As GaugeCmdTarget
	Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
	If Not MyGauge Is Nothing Then
		MyGauge.Title = "Temperature"
		Set MyGauge = Nothing
	End If
End Sub
```

TitleVisible, GaugeCmdTarget Property

Syntax Title = _Boolean

Description This property sets or returns the title's visibility within the Gauge window.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
	Dim MyGauge As GaugeCmdTarget
	Set MyGauge = GetSynopticObject.GetSubObject("MyGauge").GetObjectInterface
	If Not MyGauge Is Nothing Then
		MyGauge.TitleVisible= Not MyGauge.TitleVisible
		Set MyGauge = Nothing
	End If
End Sub
```

32. Generic Events

Click, Generic Event

Description Event occurs when the left or right mouse button is pressed within the design area.

Parameter	Description
None	None

DblClick, Generic Event

Description Event occurs when the right mouse key is double clicked within the design area. The double clicking time is set in operating system's settings.

Parameter	Description
None	None

KeyDown, Generic Event

Description Event occurs when a key is pressed down on the keyboard. This event returns the integer, KeyCode and Shift variables. This event is generated independently from being focused on.

Parameter	Description
KeyCode As Integer	Pressed Keys VBA Code. The VBA code is a set of constants which, in addition to the normal alphanumeric characters without lower/Uppercase distinction, also contemplates other keyboard keys such as the function keys, Caps Lock, etc.
Shift As Integer	Indices whether the Shift, Ctrl and Alt keys are pressed: 1 = SHIFT 2 = CTRL 4 = ALT

KeyPress, Generic Event

Description Event occurs when a key from the keyboards is pressed and released. This event returns the KeyAscii integer variable containing the pressed key's ASCII code. This event is generated only when the design is focused on.

Parameter	Description
Keyascii As Integ	The pressed key's ASCII code.

KeyUp, Generic Event

Description Event occurs when a key on the keyboard is released (after being pressed). This event releases the interger type keyCode and Shift variables. This event occurs indipendently of being focused on.

Parameter	Description
KeyCode As Integer	The pressed key's VBA code. The VBA code is a set of constants that, apart from the normal alphanumeric characters, without upper/lowercase distinction, contemplates other keys such as the Caps Lock function key etc.
Shift As Integer	Indicates whether whether the Shift, Ctrl and Alt keys are pressed: 1 = SHIFT 2 = CTRL 4 = ALT

KillFocus, Generic Event

Description Event occurs when the object in question is deselected or loses focus.

Parameter	Description
None	None

MouseDown, Generic Event

Description Event notified both in the screen code and in the object code every time the mouse key is clicked on screen, independently from its position or symbol. This event returns the integer Button and Shift type variables and the X and Y single type variables.
In order to manage this event only within a screen object you will need to use the "IsCursorOnObject" function.

Parameter	Description
Button As Integer	Indicates pressed mouse button: 1 = Left 2 = Right 4 = Central
Shift As Integer	Indicates whether the Shift, Ctrl and Alt keys are pressed: 1 = SHIFT 2 = CTRL 4 = ALT

X As Single	Horizontal coordinates referring to the cursor's position when event occurs.
Y As Single	Vertical coordinates referring to the cursor's position when event occurs.

MouseMove, Generic Event

Description Event notified both in the screen code and the object code when the mouse cursor changes position on screen, independently from the position or symbol. This event returns the Button and Shift integer type variables and the X and Y single type variables.
In order to manage this evenly only within a screen object you will need to use the "IsCursorOnObject" function.

Parameter	Description
Button As Integer	Pressed mouse key index: 1 = Left 2 = Right 4 = Central
Shift As Integer	Indicates whether the Shift, Ctrl and Alt keys are pressed: 1 = SHIFT 2 = CTRL 4 = ALT
X As Single	Horizontal coordinate referring to the cursor's positon when event occurs.
Y As Single	Vertical coordinate referring to the cursor's position when event occurs.

MouseUp, Generic Event

Description Event notified both in the screen and object codes when any one of the mouse keys are released on screen, independently from its position or symbol. This event returns the Button and Shift integer type variables and the X and Y single type variables.
In order to manage this event only within an object on screen you will need to use the "IsCursorOnObject" function.

Parameter	Description
Button As Integer	Pressed mouse key index: 1 = Left 2 = Right 4 = Central
Shift As Integer	Indicates whether the Shft, Ctrl and Alt keys are pressed: 1 = SHIFT 2 = CTRL 4 = ALT
X As Single	Horizontal coordinates referring to the cursor's positon when event occurs
Y As Single	Vertical coordinates referring to the cursor's position when event occurs

MouseWheel, Generic Events

Description Event notified when scroll is done using the mouse wheel. This event is only available for those objects that support this functionality and will be executedi if focus is on the object in question.
For further information please see paragraph "Scrolling Screen Objects With Mouse".

Parameter	Description
Button As Integer	Indicates which mouse key is pressed: 1 = Left 2 = Right 4 = Center
Shift As Integer	Indicates if the Shift, Ctrl and Alt are pressed: 1 = SHIFT 2 = CTRL 4 = ALT
Delta As Integer	Indicate in which direction the mouse wheel is rolling: if the value is higher than 0 means the scroll going upwards. If less than 0, means the scoll is going downwards.
X As Single	Horizontal coordinates referring to the cursor's position upon event.
Y As Single	Vertical coordinates referring to the cursor's position upon event.

OnChange, Generic Event

Description Event occurs when the symbol object changes its graphic status. This event returns the index relating to which graphic event changed.

The indexes are returned as follows:
enum_ONCHANGE_COMPOSEDMOVE
enum_ONCHANGE_SCALE
enum_ONCHANGE_MOVEX
enum_ONCHANGE_MOVEY
enum_ONCHANGE_TITLE
enum_ONCHANGE_STARTINGX
enum_ONCHANGE_STARTINGY
enum_ONCHANGE_ENDINGX
enum_ONCHANGE_ENDINGY
enum_ONCHANGE_FILLING
enum_ONCHANGE_ROTATION
enum_ONCHANGE_VISIBLE
enum_ONCHANGE_EDGECOLOR
enum_ONCHANGE_BACKCOLOR
enum_ONCHANGE_FILLCOLOR
enum_ONCHANGE_BITMAP
enum_ONCHANGE_XROTATIONCENTER
enum_ONCHANGE_YROTATIONCENTER

Parameter	Description
ChangeType As Integer	Graphic index change

OnChangeExecutionCanceled, Generic Event

Description Event occurs for symbol objects which try to get or move the current synapses in execution.

Parameter	Description
None	None

OnChangeExecutionToPromoter, Generic Event

Description Event occurs for symbol objects executing sysnapses which have been notified that another object is trying to change their execution flow. Setting the bRet parameter to False will stop this from happening.

Parameter	Description
None	None

OnExecutionPending, Generic Event

Description Event occurs once every second according to the symbol object executing the synapsy by indicating that the system is waiting for this object to pass this execution to another object by means of the SynapseValue, SynapseValueFromID o SynapsePassExecution functions.

Parameter	Description
None	None

OnFireExecution, Generic Event

Description Event occurs each time the design object's synapsy is about to be executed. The system establishes which design to activate the synapsy based on the logic flow represented by connections and their tabulation order. The moment the output synapsy are set in the design object with its sysnapsy in execution, the system passes the macro execution on to another design by triggering the OnFireExecution event. Therefore the logic functions concerning the handling of the symbol's input and output synapsy are inserted in this event.

Parameter	Description
None	None

OnFireSynapse, Generic Event

Description This event is generated every time an input synapse receives the value from an output synapse linked to it, or in other words then a drawing object has set the value of the output synapse, by means of using the SynapseValue, SynapseValueFromID or SynapsePassExecution properties to which the input synapse, receiving the event, is linked.
You can find out which input synapse has been effected by the event described by using the SynapseName parameter.

Parameter	Description
SynapseName As String	Name of the synapse which has just received the value.

OnGesture, Generic Event

Description This event is notified when a Gesture movement is made in the project.

Parameter	Description
Button As Integer	Press mouth key index: 1 = Left 2 = Right 4 = Central
Shift As Integer	Indices if the Shift, Ctrl and Alt keys are pressed: 1 = SHIFT 2 = CTRL 4 = ALT
X1 As Integer	Horizontal coordinates refer to the cursor's position at beginning of movement.
Y1 As Integer	Vertical coordinates refer to the cursor's position at beginning of movement.
X2 As Integer	Horizontal coordinates refer to the cursor's position at end of movement.
Y2 As Integer	Vertical coordinates refer to the cursor's position at end of movement.
ElapsedTime As Single	Elasped time of Gesture movement.
bRet as Boolean	When this parameter is set at True, the Gesture movement event will be aborted and no operation will be executed.

OnPostPaint, Generic Event

Description Event occurs each time the design object gets its graphics refreshed by the system. This can happen under different circumstances, for example following the opening of a screen, when the application is focused on, and every time the design's animation is executed or its position recalculated on screen.
The hde parameter (Handle to the device context) gieves useful information for the expert Windows user on the graphic refresh types adopted by Movicon.

Parameter	Description
ByVal hdc As Long	Handle to the device context.

OnPrePaint, Generic Event

Description Event occurs each time the design object gets its graphics refreshed by the system. This can happen under different circumstances, for example following the opening of a screen, when the application is focused on, and every time the design's animation is executed or its position recalculated on screen.

The symbol's graphic refresh is disabled when the bRet parameter is set to false.
The hde parameter (Handle to the device context) gieves useful information for the expert Windows user on the graphic refresh types adopted by Movicon.

Parameter	Description
ByVal hdc As Long	Handle to the device context
bRet As Boolean	Enable redesign

OnPreSymbolLoading, Generic Event

Description This event is called before the object is initialized, therefore also before the "SymbolLoading" event and permits those operations to be carried out which otherwise would be ignored because the object had already been initialized. For example, the default structure or a component within a symbol can be changed and set with the object's public name.

After the "OnPreSymbolLoading" event has been loaded, the object's script code is destroyed (as if it had never been executed). This means that, for example, the creation of object type variables within the event will no longer be valid once the symbols had been loaded.

The "OnPreSymbolLoading" event is NEVER called in objects contained within Embedded Screen or Tab Groups.

Parameter	Description
None	None

OnSize, Generic Event

Description This event verifies when the object is resized to runtime, due to the resizing of the screen or the object itself when enabled with the Drag/Resize options.

Parameter	Description
nWidth As Integer	Out parameter. Returns object to width in pixels.
nHeight As Integer	Out parameter. Returns object to height in pixels.

Example:

```
Public Sub OnSize(ByRef nWidth As Integer, ByRef nHeight As Integer)
        MsgBox "object width = " & nWidth & ",  object height =  " & nHeight
End Sub
```

OnTextChanged, Generic Event

Description Event occurs when the text of the object's title has been changed with the keyboard. The ChangedText string varialbe containing the new text is returned.

In cases where the project's password managment has been enabled, the "OnTextChanged" event in objects will automatically request for user login in accordance to the password level set in that object.

Parameter	Description
ChangedText As String	New text containing the Title field.

OnTextChanging, Generic Event

Description Event occurs when the object's title is changed with the keyboard. The bRet boolean variable allows or does not allow this change: when the bRet variable is set to False within the function, the changes made to the text contained in the object will have no effect.

Parameter	Description
bRet As Boolean	Text changing enabling

OnTimer, Generic Event

Description Event occurs with a period of about 1/2 seconds (time not guaranteed) during runtime mode. During the Test mode this period is proportional to

the set test velocity. The event's execution time can be customized by means of the TimerEventFrequence registry key.

Parameter	Description
None	None

OnToolTip, Generic Event

Description Event occurs when the obnject is in a condition to display a "pop-up" string called ToolTip (eg. pointed by mouse). the Show boolean variable is returned True when the mouse is on the object and False when the mouse is outside the object. To display th ToolTip you must set the relevant method.

Parameter	Description
Show As Boolean	Variable which indicates whether the mouse cursor is on the object.

SetFocus, Generic Event

Description Event occurs when the design object receives focus or is selected.

Parameter	Description
None	None

SymbolLoading, Generic Event

Description Event notified when the drawing object is loaded in memory, therefore at the opening of the screen it belongs to. This event is independent of the design's visibility conditions.

Parameter	Description
None	None

SymbolUnloading, Generic Event

Description Event occurs when the design object is unloaded from memory when the screen closes. This event is indiependent of the design's visibility conditions.

Parameter	Description
None	None

33. GridWndCmdTarget

OnColumnClick, GridWndCmdTarget Event

Description Event occurs each tmie the mouse key is pressed on a grid cell.

Parameter	Description
nRow As Integer	Cell's row number.
nCol As Integer	Cell's column number.

OnDeleteItem, GridWndCmdTarget Event

Description Event occurs each time a grid element is deleted.

Parameter	Description
nRow As Integer	Cell's row number.
nCol As Integer	Cell's column number.

OnQueryEnd, GridWndCmdTarget Event

Description Event occurs at the end of each data extraction with a SQL selection query.

Parameter	Description
None	None

OnQueryNext, GridWndCmdTarget Event

Description Event occurs at the end of the extraction of each record from the associated database.

Parameter	Description

bRet As Boolean	function execution enabling

OnQueryStart, GridWndCmdTarget Event

Description Event occurs at the extraction start of each data extraction with the SQL selection query.

Parameter	Description
bRet As Boolean	Function execution enabling

OnSelChanged, GridWndCmdTarget Event

Description Event occurs each time the selection of the cell in the grid is changed.

Parameter	Description
nRow As Integer	Selected cell's row number
nCol As Integer	Selected cell's column number.

OnSelChanging, GridWndCmdTarget Event

Description Event occurs each time the cell selection in the grid is about to change.

Parameter	Description
nRow As Integer	Selected cell's row number
nCol As Integer	Selected cell's column number

OnSQLError, GridWndCmdTarget Event

Description Event occurs each time an error is generated while extracting data from the database.

Parameter	Description
Error As String	String containing the Database manager error.

OnUpdatingDSN, GridWndCmdTarget Event

Description Event occurs each time a request is made with the 'Save' command for edits (INSERT\DELETE\UPDATE) made to the database linked to the grid.

Parameter	Description
SQLCommand As String	String containing the SQL command to be executed
bRet As Boolean	Enable command execution

AddColumn, GridWndCmdTarget Function

Syntax AddColumn(_lpszColumnName)

Description This function allows you to inset a new column in grid object. The parameter contains the name of the column to be inserted.

Parameter	Description
lpszColumnName As String	Name of the column to be inserted.

Result None

Example:

```
Dim objGrid As GridWndCmdTarget
Public Sub Click()
        objGrid.AutoLayout = True
        objGrid.AddColumn("New column")
        objGrid.RecalcLayout
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

CellEditable, GridWndCmdTarget Function

Syntax CellEditable(_nRow,_nCol)

Description This function returns the True Boolean value when the contents of the cell referenced by the row and column number is editable.

Parameter	Description
nRow As Long	Row number

nCol As Long	Column number

Result Boolean

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
Public Sub Click()
	If objGrid.CellEditable(1,1) Then
		MsgBox("Is editable!!",vbOkOnly,GetProjectTitle)
	Else
		MsgBox("Is Not editable!!",vbOkOnly,GetProjectTitle)
	End If
End Sub
Public Sub SymbolLoading()
	Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

DeleteColumn, GridWndCmdTarget Function

Syntax DeleteColumn(_lpszColumnName)

Description This function allows you to delete a column from the grid object. The parameter contains the name of the column to be deleted. This function returns the True boolean value when the delete operation is successful.

Parameter	Description
lpszColumnName As String	Name of the column to be deleted

Result None

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
Public Sub Click()
	objGrid.AutoLayout = True
	Debug.Print objGrid.DeleteColumn("Colonna10")
	objGrid.RecalcLayout
End Sub
Public Sub SymbolLoading()
	Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

DeleteRow, GridWndCmdTarget Function

Syntax DeleteRow()

Description This function deletes the row selected from the grid.

Parameter	Description
None	None

Result Boolean

Example:

```
Dim objGrid As GridWndCmdTarget
Public Sub Click()
        objGrid.DeleteRow
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

EditCopy, GridWndCmdTarget Function

Syntax EditCopy()

Description This property returns the True boolean value when the data contained in the cell referenced by the row and column number passed as parameter has been changed.

This function is not supported Windows CE.

Parameter	Description
None	None

Result None

Example:

```
'Button environment
Option Explicit
Dim objGrid As GridWndCmdTarget
Dim X
Public Sub Click()
        objGrid.SetSelectedRange(1,1,1,3)
        objGrid.EditCopy
        Debug.Print Clipboard$()
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

EnsureVisible, GridWndCmdTarget Function

Syntax EnsureVisible()

Description This function forces the Grid Window's scroll to show the cell referenced by the row and column number passed as parameters.

Parameter	Description
None	None

Result None

Example:

```
'Button environment
Option Explicit
Dim objGrid As GridWndCmdTarget
Dim X
Public Sub Click()
        objGrid.EnsureVisible(3,3)
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

FocusCellEditable, GridWndCmdTarget Function

Syntax FocusCellEditable()

Description This function returns the True boolean value when the cell contents are editable.

Parameter	Description
None	none

Result Boolean

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
Public Sub Click()
        Debug.Print objGrid.FocusCellEditable
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

GetColCount, GridWndCmdTarget Function

Syntax GetColCount()

Description This function gets the number of columns loaded into the Grid object.

Parameter	Description
None	None

Result Long

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
Public Sub Click()
        Debug.Print objGrid.GetColCount
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

GetRowCount, GridWndCmdTarget Function

Syntax GetRowCount()

Description This function get the number of rows loaded into the grid object. The number of rows correspond to the number of records retrieved by the selection query when connected to a database, or the number of rows existing in a linked text file.

Parameter	Description
None	None

Result Long

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
Public Sub Click()
        Debug.Print objGrid.GetRowCount
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

GetSelectedRange, GridWndCmdTarget Function

Syntax GetSelectedRange(_nMinRow, _nMinCol, _nMaxRow, _nMaxCol)

Description This function is used for retrieving the row, start and end column numbers of a multi-cell selection.

Parameter	Description
nMinRow as long	Selection's start row number.
nMinCol as long	Selection's start column number.
nMaxRow as long	Selection's end row number.
nMaxCol as long	Selection's end column number.

Result Long

Example:

```
Public Sub Click()
        Dim objGrid As GridWndCmdTarget
        Dim nMinRow As Long
        Dim nMinCol As Long
        Dim nMaxRow As Long
        Dim nMaxCol As Long

        Set objGrid = GetSynopticObject.GetSubObject("objGrid").GetObjectInterface
        objGrid.GetSelectedRange(nMinRow, nMinCol, nMaxRow, nMaxCol)

        MsgBox "Grid Selected Range:" & vbLf & _
        "MinRow = " & nMinRow & vbLf & _
        "MinCol = " & nMinCol & vbLf & _
        "MaxRow = " & nMaxRow & vbLf & _
        "MaxCol = " & nMaxCol, vbInformation, GetProjectTitle

        Set objGrid = Nothing
End Sub
```

EditCopy, GridWndCmdTarget Function

Syntax EditCopy()

Description This property returns the True boolean value when the data contained in the cell referenced by the row and column number passed as parameter has been changed.

This function is not supported Windows CE.

Parameter	Description
None	None

Result None

Example:

```
'Button environment
Option Explicit
Dim objGrid As GridWndCmdTarget
Dim X
Public Sub Click()
        objGrid.SetSelectedRange(1,1,1,3)
        objGrid.EditCopy
        Debug.Print Clipboard$()
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

EnsureVisible, GridWndCmdTarget Function

Syntax EnsureVisible()

Description This function forces the Grid Window's scroll to show the cell referenced by the row and column number passed as parameters.

Parameter	Description
None	None

Result None

Example:

```
'Button environment
Option Explicit
Dim objGrid As GridWndCmdTarget
Dim X
Public Sub Click()
        objGrid.EnsureVisible(3,3)
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

FocusCellEditable, GridWndCmdTarget Function

Syntax FocusCellEditable()

Description This function returns the True boolean value when the cell contents are editable.

Parameter	Description
None	none

Result Boolean

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
Public Sub Click()
        Debug.Print objGrid.FocusCellEditable
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

GetColCount, GridWndCmdTarget Function

Syntax GetColCount()

Description This function gets the number of columns loaded into the Grid object.

Parameter	Description
None	None

Result Long

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
Public Sub Click()
        Debug.Print objGrid.GetColCount
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

GetRowCount, GridWndCmdTarget Function

Syntax GetRowCount()

Description This function get the number of rows loaded into the grid object. The number of rows correspond to the number of records retrieved by the selection query when connected to a database, or the number of rows existing in a linked text file.

Parameter	Description
None	None

Result Long

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
Public Sub Click()
	Debug.Print objGrid.GetRowCount
End Sub
Public Sub SymbolLoading()
	Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

GetSelectedRange, GridWndCmdTarget Function

Syntax GetSelectedRange(_nMinRow, _nMinCol, _nMaxRow, _nMaxCol)

Description This function is used for retrieving the row, start and end column numbers of a multi-cell selection.

Parameter	Description
nMinRow as long	Selection's start row number.
nMinCol as long	Selection's start column number.
nMaxRow as long	Selection's end row number.
nMaxCol as long	Selection's end column number.

Result Long

Example:

```
Public Sub Click()
	Dim objGrid As GridWndCmdTarget
	Dim nMinRow As Long
	Dim nMinCol As Long
	Dim nMaxRow As Long
	Dim nMaxCol As Long

	Set objGrid = GetSynopticObject.GetSubObject("objGrid").GetObjectInterface
	objGrid.GetSelectedRange(nMinRow, nMinCol, nMaxRow, nMaxCol)

	MsgBox "Grid Selected Range:" & vbLf & _
	"MinRow = " & nMinRow & vbLf & _
	"MinCol = " & nMinCol & vbLf & _
	"MaxRow = " & nMaxRow & vbLf & _
	"MaxCol = " & nMaxCol, vbInformation, GetProjectTitle

	Set objGrid = Nothing
End Sub
```

InsertRow, GridWndCmdTarget Function

Syntax InsertRow()

Description This function inserts Row at the bottom of the grid.

Parameter	Description
None	None

Result Boolean

Example:

```
Dim objGrid As GridWndCmdTarget
Public Sub Click()
	objGrid.InsertRow
End Sub
Public Sub SymbolLoading()
	Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

IsCellSelected, GridWndCmdTarget Function

Syntax IsCellSelected(_nRow,_nCol)

Description This function returns the True boolean value if the cell, referenced by the row and column number, is selected.

Parameter	Description
nRow As Long	Row number
nCol As Long	Column number

Result Boolean

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
Public Sub Click()
	NumericEntry("riga")
	NumericEntry("colonna")
	If objGrid.IsCellSelected(riga,colonna) Then
		MsgBox("Is selected!!",vbOkOnly,GetProjectTitle)
	Else
		MsgBox("Is NOT selected!!",vbOkOnly,GetProjectTitle)
	End If
End Sub
Public Sub SymbolLoading()
	Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

IsCellValid, GridWndCmdTarget Function

Syntax IsCellValid(_nRow,_nCol)

Description This function returns the True boolean value if the cell, referenced by the row and column number, exists in the grid window.

Parameter	Description
nRow As Long	Row number
nCol As Long	Column number

Result Boolean

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
Public Sub Click()
	NumericEntry("riga")
	NumericEntry("colonna")
	If objGrid.IsCellValid(riga,colonna) Then
		MsgBox("Is valid!!",vbOkOnly,GetProjectTitle)
	Else
		MsgBox("Is NOT valid!!",vbOkOnly,GetProjectTitle)
	End If
End Sub
Public Sub SymbolLoading()
	Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

IsCellVisible, GridWndCmdTarget Function

Syntax IsCellVisible(_nRow,_nCol)

Description This function returns the True boolean value if the cell, referenced by the row and column number, is visible on screen.

Parameter	Description
nRow As Long	Row number
nCol As Long	Column number

Result Boolean

Example:

```
'Button environment
```

```
Dim objGrid As GridWndCmdTarget
Public Sub Click()
        NumericEntry("riga")
        NumericEntry("colonna")
        If objGrid.IsCellVisible(riga,colonna) Then
                MsgBox("Is visible!!",vbOkOnly,GetProjectTitle)
        Else
                MsgBox("Is NOT visible!!",vbOkOnly,GetProjectTitle)
        End If
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

LoadFromTextFile, GridWndCmdTarget Function

Syntax LoadFromTextFile()

Description This function is used for loading data from a text file in the grid. The text file must be saved in unicode format.

Parameter	Description
None	None

Result Boolean

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
Public Sub Click()
        objGrid.TextFileName = MyProjectPath & "\data\prodotti.txt"
        objGrid.LoadFromTextFile
        objGrid.Refresh
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

LoadExtSettings, GridWndCmdTarget Function

Syntax LoadExtSettings

Description This function permits the object's relating external file settings to be loaded. This file can be specified in design mode in the "External File settings" property or in the "ExtSettingsFile" interface properties. The extension provided for this file is ".SXML".

Parameter	Description

None	None

Result Boolean

Example:

```
Public Sub Click()
Dim objSymbol As GridWndCmdTarget
Set objSymbol = GetSynopticObject.GetSubObject("TestObject").GetObjectInterface
If objSymbol Is Nothing Then Exit Sub
objSymbol.ExtSettingsFile = "test.sxml"
objSymbol.LoadExtSettings
Set objSymbol = Nothing
End Sub
```

RecalcLayout, GridWndCmdTarget Function

Syntax RecalcLayout()

Description This function recalculates the object's graphical layout. This function must be executed after a change has been made to the properties concerning the object's layout such as the AutoLayout property described in this section.

Parameter	Description
None	None

Result None

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
Public Sub Click()
        objGrid.AutoLayout = Not(objGrid.AutoLayout)
        objGrid.RecalcLayout
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

Refresh, GridWndCmdTarget Function

Syntax Refresh()

Description This function refreshes the data displayed in the grid according to the set query.

Parameter	Description
None	none

Result Boolean

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
Public Sub Click()
        objGrid.Refresh
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

SaveToTextFile, GridWndCmdTarget Function

Syntax SaveToTextFile()

Description This function is used for saving data in the grid on text files. The text file will be unicode format.

Parameter	Description
None	None

Result Boolean

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
Public Sub Click()
        objGrid.TextFileName = MyProjectPath & "\data\prodotti.txt"
        objGrid.LoadFromTextFile
        objGrid.CellText(1,0) = "Cambio Testo"
        objGrid.Refresh
        objGrid.SaveToTextFile
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

SaveExtSettings, GridWndCmdTarget Function

Syntax SaveExtSettings

Description This function permits the objects settings to be save in the relating external settings file. This file can be specified when in design mode in the "Ext. Settings File" property, or using the property from the "ExtSettingsFile" interface. The extension provided for this file is ".SXML".

Parameter	Description

None	None

Result Long

Example:

```
Public Sub Click()
Dim objSymbol As GridWndCmdTarget
Set objSymbol = GetSynopticObject.GetSubObject("TestObject").GetObjectInterface
If objSymbol Is Nothing Then Exit Sub
objSymbol.ExtSettingsFile = "test.sxml"
objSymbol.SaveExtSettings
Set objSymbol = Nothing
End Sub
```

SelectAll, GridWndCmdTarget Function

Syntax SelectAll()

Description This function allows to select all the grid object's cells.

Parameter	Description
None	None

Result None

Example:

```
'Button environment
Option Explicit
Dim objGrid As GridWndCmdTarget
Dim X
Public Sub Click()
        objGrid.SelectAll
        objGrid.EditCopy
        MsgBox Clipboard$(),vbOkOnly,GetProjectTitle
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

SetSelectedRange, GridWndCmdTarget Function

Syntax SetSelectedRange(_nMinRow, _nMinCol, _nMaxRow, _nMaxCol)

Description This function allows you to select all the grid object's cells within the range passed as parameter.

Parameter	Description
nMinRow As Long	Start row number
nMinCol As Long	Start column number
nMaxRow As Long	End row number
nMaxCol As Long	End column number

Result None

Example:

```
'Button environment
Option Explicit
Dim objGrid As GridWndCmdTarget
Dim X
Public Sub Click()
        objGrid.SetSelectedRange(1,1,3,3)
        objGrid.EditCopy
        MsgBox Clipboard$(),vbOkOnly,GetProjectTitle
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

UpdateDatabase, GridWndCmdTarget Function

Syntax UpdateDatabase()

Description This function saves data on the database relating to the cells changed by using codes (with the FocusCellText function) or the keyboard. This function returns the True boolean value when the save has been done.

Parameter	Description
None	none

Result Boolean

Example:

```
Dim objGrid As GridWndCmdTarget
Public Sub Click()
        objGrid.FocusCellCol = 1
        objGrid.FocusCellRow = 1
        objGrid.FocusCellText = "text1"
        objGrid.RcalcLayout
        objGrid.UpdateDatabase
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

UpdateVariables, GridWndCmdTarget Function

Syntax UpdateVariables()

Description This function allows the variables to be updated with the data in the grid's columns with the same name. The data with which the variables are updated corresponds to the row selected. This function has effect only when the UpdateVariable property, described in this section, is set with the True boolean value.

Parameter	Description
None	none

Result Boolean

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
Public Sub Click()
        If objGrid.UpdateVariable Then
                objGrid.UpdateVariables
        End If
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

AutoLayout, GridWndCmdTarget Property

Syntax AutoLayout = _Boolean

Description When this property is enabled, the list layout will be set automatically. This means that the table's columns will automatically resize to fit within the Viewer window area. When this property is disabled the columns will maintain the sizes they were set with in the programming stage when the window is opened. This may not allow for all the columns to fit in the window and therefore the last ones on the right will have to be viewed by using the horizontal scroll bar.

Parameter	Description
None	none

Result Boolean

Example:

```
Dim objGrid As GridWndCmdTarget
```

```
Public Sub Click()
        objGrid.AutoLayout = Not(objGrid.AutoLayout)
        objGrid.RecalcLayout
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

ButtonPos, GridWndCmdTarget Property

Syntax ButtonPos = _Integer

Description This property sets or returns the position of the buttons in the window.

The possible positions are:
enum_gba_left
enum_gba_top
enum_gba_right
enum_gba_bottom

Parameter	Description
None	none

Result Integer

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
Public Sub Click()
        Dim pos
        pos = objGrid.ButtonPos
        Select Case size
                Case enum_gba_left
                Debug.Print "Pos = Left"
                Case enum_gba_top
                Debug.Print "Pos = Top"
                Case enum_gba_right
                Debug.Print "Pos = Right"
                Case enum_gba_bottom
                Debug.Print "Pos = Bottom"
                Case Else
        End Select
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

ButtonSize, GridWndCmdTarget Property

Syntax ButtonSize = _Integer

Description This property sets or returns the size of the buttons in the window.

The sizes can be:
enum_gbz_small
enum_gbz_medium
enum_gbz_large

Parameter	Description
None	none

Result Integer

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
Public Sub Click()
        Dim size
        size = objGrid.ButtonSize
        Select Case size
                Case enum_gbz_small
                Debug.Print "Size = Small"
                Case enum_gbz_medium
                Debug.Print "Size = Medium"
                Case enum_gbz_large
                Debug.Print "Size = Large"
                Case Else
        End Select
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

CellBkColor, GridWndCmdTarget Property

Syntax CellBkColor(_nRow,_nCol)

Description This property sets or returns the back color of the Cell referenced by the row number in the column passed as parameter.

Cell coloring only works correctly if the Grid is set to display rows with one unique colour, being that the "RowsOneColor" registry key is set at "1". Otherwise the grid row colours will obtain different colours to those set with the "CellBkColor" property.

Parameter	Description
nRow As Long	Row number
nCol As Long	Column number

Result Long

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
Public Sub Click()
        objGrid.CellBkColor(1,1) = RGB(255,0,0)
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

CellFgColor, GridWndCmdTarget Property

Syntax CellFgColor(_nRow,_nCol)

Description This property sets or returns the text color of the cell identified by the row and column number passed as parameter.

Parameter	Description
nRow As Long	Row number
nCol As Long	Column number

Result Long

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
Public Sub Click()
        objGrid.CellFgColor(1,1) = RGB(255,0,0)
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

CellModified, GridWndCmdTarget Property

Syntax CellModified(_nRow,_nCol) = _Boolean

Description This property returns or sets the True Boolean value when the data contained in the cell referenced by the row number and column number passed as parameter has been changed. All the cells start with the "modified" property set at false when data is loaded (upon page opening or with the refresh method). When a change is made from the keyboard or with a code, this property is set at true. This property can be set at false before executing a refresh to check whether data has been loaded effectively.

Parameter	Description
nRow As Long	Row number
nCol As Long	Column number

Result Boolean

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
Public Sub Click()
	If objGrid.CellModified(1,1) Then
		MsgBox("Has been modified!!",vbOkOnly,GetProjectTitle)
		objGrid.CellModified(1,1) = False
	End If
End Sub
Public Sub SymbolLoading()
	Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

CellText, GridWndCmdTarget Property

Syntax CellText(_nRow,_nCol)

Description This property sets or returns the text contained in the cell referenced by the row number and column number passed as parameter.

Parameter	Description
nRow As Long	Row number
nCol As Long	Column number

Result String

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
Public Sub Click()
	Debug.Print objGrid.CellText(1,1)
End Sub
Public Sub SymbolLoading()
	Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

Clickable, GridWndCmdTarget Property

Syntax Clickable = _Boolean

Description When this property is set to False, it will no longer be possible to manage the control with the mouse or the keyboard. Therefore is will be impossible to put the columns into order, view help, execute and commands in the window.

Parameter	Description

None	none

Result Boolean

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
Public Sub Click()
        Debug.Print objGrid.Clickable
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

ColumnWidths, GridWndCmdTarget Property

Syntax ColumnWidths = _String

Description This property is used for reading or setting Grid column widths in pixels. This property is a string in which values should be entered indicating the sizes of each column separated by the pipe (|) character. Each value will be associated to the column based in order of sequence.

Parameter	Description
None	none

Result String

Example:

```
Dim objGrid As GridWndCmdTarget
Public Sub Click()
        Debug.Print objGrid.ColumnWidths
End Sub
Public Sub SymbolLoading()
        Set objGrid =
        GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

CopyBtnText, GridWndCmdTarget Property

Syntax CopyBtnText = _String

Description This property sets or returns any customized text to be displayed in the 'Copy' button.

This property is not supported in Windows CE. (If set, always returns an empty string)

Parameter	Description

None	none

Result String

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
Public Sub Click()
        Debug.Print objGrid.CopyBtnText
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

DeleteBtnText, GridWndCmdTarget Property

Syntax DeleteBtnText = _String

Description This property sets or returns any customized text to be viewed in the 'Delete' button.

Parameter	Description
None	none

Result String

Example:

```
'Environment button
Dim objGrid As GridWndCmdTarget
Public Sub Click()
        Debug.Print objGrid.DeleteBtnText
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

DSN, GridWndCmdTarget Property

Syntax DSN = _String

Description This property allows you to read or set the name of the **ODBC** connection for the Grid object.

Parameter	Description
None	none

Result String

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
Public Sub Click()
        Debug.Print objGrid.DSN
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

ExtSettingsFile, GridWndCmdTarget Property

Syntax ExtSettingsFile = _String

Description This property sets or returns the external configuration file for the referenced object. the file can be alsow specified in design mode in the object's "Configuration File" property. The extension provided for this file is ".SXML".

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
Dim objSymbol As GridWndCmdTarget
Set objSymbol = GetSynopticObject.GetSubObject("TestObject").GetObjectInterface
If objSymbol Is Nothing Then Exit Sub
objSymbol.ExtSettingsFile = "test.sxml"
objSymbol.SaveExtSettings
Set objSymbol= Nothing
End Sub
```

FocusCellBkColor, GridWndCmdTarget Property

Syntax FocusCellBkColor = _Long

Description This property sets or returns the back color of the selected cell.

Parameter	Description
None	none

Result Long

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
Public Sub Click()
        objGrid.FocusCellBkColor = RGB(255,0,0)
        objGrid.RecalcLayout
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

FocusCellCol, GridWndCmdTarget Property

Syntax FocusCellCol = _Long

Description This property set or returns the number related to the column of the cell selected. Returns the -1 value when no cells have been selected.

Parameter	Description
None	none

Result Long

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
Public Sub Click()
        Debug.Print objGrid.FocusCellCol
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

FocusCellFgColor, GridWndCmdTarget Property

Syntax FocusCellFgColor = _Long

Description This property sets or returns the text color of the selected cell.

Parameter	Description
None	none

Result Long

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
Public Sub Click()
        objGrid.FocusCellFgColor = RGB(255,0,0)
        objGrid.RecalcLayout
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

FocusCellModified, GridWndCmdTarget Property

Syntax FocusCellModified = _Boolean

Description This property returns the True Boolean value if the cell's contents have been changed. All cells start with the "modified" property at false when data is loaded (when a page is opened or with the refresh method). When any changes are made with the keyboard or with codes, the property is placed at true. This property can be set at false before a refresh is carried out for checking whether data has been loaded effectively.

Parameter	Description
None	none

Result Boolean

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
Public Sub Click()
        Debug.Print objGrid.FocusCellModified
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

FocusCellRow, GridWndCmdTarget Property

Syntax FocusCellRow = _Long

Description This property set or returns the number relating to the row of the selected cell. Returns the -1 value if no cells have been selected.

Parameter	Description

None	none

Result Long

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
Public Sub Click()
	Debug.Print objGrid.FocusCellRow
End Sub
Public Sub SymbolLoading()
	Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

FocusCellText, GridWndCmdTarget Property

Syntax FocusCellText = _String

Description This property returns the text of the selected cell. Use the UpdateDatabase function to update the database after any modifications have been carried out.

Parameter	Description
None	none

Result String

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
Public Sub Click()
	Debug.Print objGrid.FocusCellText
End Sub
Public Sub SymbolLoading()
	Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

GraphicButtons, GridWndCmdTarget Property

Syntax GraphicButtons = _Boolean

Description When Enabling this property, the Grid Window buttons are drawn using an icon instead of text. The text will instead be displayed as a tooltip when positioning the mouse on top of the button.

The tooltip is not managed in Windows CE versions.

This property is not managed by the 'Alarm Banner' object.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Click()
        GraphicButtons = True
        RecalcLayout
End Sub
```

InsertBtnText, GridWndCmdTarget Property

Syntax InsertBtnText = _String

Description This property sets or returns any customized text to be viewed in the 'Insert' button.

Parameter	Description
None	none

Result String

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
Public Sub Click()
        Debug.Print objGrid.InsertBtnText
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

Promptpad,GridWndCmdTarget_Property

Syntax PromptPad = _Boolean

Description When setting this property to "True", the alphanumeric Pad will be used for editing grid values. When setting it to "False", the values will be editable directly in the grid cells.

Parameter	Description
None	none

Result Boolean

Example:

```
Public Sub Click()
        Dim objGrid As GridWndCmdTarget
        Set objGrid = GetSynopticObject.GetSubObject("grid1").GetObjectInterface
        If objGrid Is Nothing Then Exit Sub
        If objGrid.PromptPad = False Then
                objGrid.PromptPad = True
        End If
End Sub
```

Query, GridWndCmdTarget Property

Syntax Query = _String

Description This property sets or returns the SQL standard query language on data to be selected from the Database.

Parameter	Description
None	none

Result String

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
Public Sub Click()
        Debug.Print objGrid.Query
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

SaveBtnText, GridWndCmdTarget Property

Syntax SaveBtnText = _String

Description This property sets or returns any customized text to be displayed in the 'Save' key.

Parameter	Description
None	none

Result String

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
Public Sub Click()
        Debug.Print objGrid.SaveBtnText
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

SelectAllBtnText, GridWndCmdTarget Property

Syntax SelectAllBtnText = _String

Description This property sets or returns any customized text to be displayed in the 'Select All' button.

Parameter	Description
None	none

Result String

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
Public Sub Click()
        Debug.Print objGrid.SelectAllBtnText
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

ShowCopyBtn, GridWndCmdTarget Property

Syntax ShowCopyBtn = _Boolean

Description This property sets or returns the possibility to show the 'Copy' button in the grid window. The button is displayed in the grid when the boolean value is set at true.

This property is not supported in Windows CE.(If set, always returns 'false')

Parameter	Description
None	none

Result Boolean

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
Public Sub Click()
        Debug.Print objGrid.ShowCopyBtn
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

ShowDeleteBtn, GridWndCmdTarget Property

Syntax ShowDeleteBtn = _Boolean

Description This property sets or returns the possibility to show the 'Delete' button in the grid window. The button is displayed in the grid when the boolean value is set at true.

Parameter	Description
None	none

Result Boolean

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
Public Sub Click()
        Debug.Print objGrid.ShowDeleteBtn
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

ShowInsertBtn, GridWndCmdTarget Property

Syntax ShowInsertBtn = _Boolean

Description This property sets or returns the possibility to show the Insert' button in the grid window for inserting rows. The button is displayed in the grid when the boolean value is set at true.

Parameter	Description
None	none

Result Boolean

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
```

```
Public Sub Click()
        Debug.Print objGrid.ShowInsertBtn
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

ShowSaveBtn, GridWndCmdTarget Property

Syntax ShowSaveBtn = _Boolean

Description This property sets or returns the possibility to show the 'Save' button in the grid window. The button is displayed in the grid when the boolean value is set at true.

Parameter	Description
None	none

Result Boolean

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
Public Sub Click()
        Debug.Print objGrid.ShowSaveBtn
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

ShowSelectAllBtn, GridWndCmdTarget Property

Syntax ShowSelectAllBtn = _Boolean

Description This property sets or returns the possibility to show the 'Select All' button in the grid window. This button is displayed in the grid to select all data when the boolean value is set at true.

Parameter	Description
None	none

Result Boolean

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
Public Sub Click()
```

```
        Debug.Print objGrid.ShowSelectAllBtn
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

ShowUpdateBtn, GridWndCmdTarget Property

Syntax ShowUpdateBtn = _Boolean

Description This property sets or returns the possibility to show the 'Update' button in the grid window. This button is displayed in the grid to save data in the grid when the boolean value is set at true.

Parameter	Description
None	none

Result Boolean

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
Public Sub Click()
        Debug.Print objGrid.ShowUpdateBtn
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

TextFileName, GridWndCmdTarget Property

Syntax TextFileName = _String

Description This property sets or returns the name of the text file (saved in unicode format) containing the data needed to fill the grid.
CAUTION: When this property contains a file name, the Grid object will get data from that file and not from any exiting ODBC link defined in the object. In order to get the Grid object to work with the ODBC link again, you must reset the "TextFileName" property null value (TextFileName = "").

Parameter	Description
None	none

Result String

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
```

```
Public Sub Click()
        Debug.Print objGrid.TextFileName
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

TextSeparator, GridWndCmdTarget Property

Syntax TextSeparator = _String

Description This property sets or returns the text separator between one set of data and the next within the text file (leave this field empty to use the ',' default separator). The text must be saved in unicode format.

Parameter	Description
None	none

Result String

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
Public Sub Click()
        Debug.Print objGrid.TextSeparator
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

UpdateBtnText, GridWndCmdTarget Property

Syntax UpdateButtonText = _String

Description This property sets or returns any customized text to be displayed in the 'Update' key.

Parameter	Description
None	none

Result String

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
Public Sub Click()
        Debug.Print objGrid.UpdateButtonText
End Sub
Public Sub SymbolLoading()
```

```
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

UpdateVariable, GridWndCmdTarget Property

Syntax UpdateVariable = _Boolean

Description When this property is set at True the variable will be updated with the same data of the columns. The effective update is carried out by calling the UpdateVariables function. The data with which the variables are updated corresponds to the data of the row selected.

Parameter	Description
None	none

Result Boolean

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
Public Sub Click()
        objGrid.UpdateVariable = True
        objGrid.UpdateVariables
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

UserName, GridWndCmdTarget Property

Syntax UserName = _String

Description This property specified the name of the user used for the **ODBC** connection.

Parameter	Description
None	none

Result String

Example:

```
'Button environment
Dim objGrid As GridWndCmdTarget
Public Sub Click()
        Debug.Print objGrid.UserName
End Sub
Public Sub SymbolLoading()
        Set objGrid = GetSynopticObject.GetSubObject("GridWindow").GetObjectInterface
End Sub
```

34. HisLogWndCmdTarget

OnFilter, HisLogWndCmdTarget Event

Description Event occurs each time a request is made to apply a filter for extracting data from the historical Log.

Parameter	Description
bRet As Boolean	Enable upon status change.

OnPrint, HisLogWndCmdTarget Event

Description Event occurs each time a request is made to print data loaded in the display window.

This event is not supported in Windows CE.

Parameter	Description
bRet As Boolean	Enable to send print.

OnRefresh, HisLogWndCmdTarget Event

Description Event occurs each time a request is made to refresh data loaded in the display window.

Parameter	Description
bRet As Boolean	Enable upon status change.

EditCopy, HisLogWndCmdTarget Function

Syntax EditCopy()

Description This property copies the selected line's contents on the clipboard.

Parameter	Description

None	None

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
	Dim HisWindow As HisLogWndCmdTarget
	Set HisWindow = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
	If Not HisWindow Is Nothing Then
		HisWindow.EditCopy
	End If
	Set HisWindow = Nothing
End Sub
```

EditLayout, HisLogWndCmdTarget Function

Syntax EditLayout()

Description This function opens the configuration window of fields to be displayed in the Historical Log Window.

This function is only executed if the "Show Control window" property has been enabled in the Window object. Otherwise the "Field Choice Window" will not open and this function will return the "False" value.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
	Dim HisWindow As HisLogWndCmdTarget
	Set HisWindow = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
	If Not HisWindow Is Nothing Then
		HisWindow.EditLayout
	End If
	Set HisWindow = Nothing
End Sub
```

LoadExtSettings, HisLogWndCmdTarget Function

Syntax LoadExtSettings

Description This function permits the object's relating external file settings to be loaded. This file can be specified in design mode in the "External File settings" property or in

the "ExtSettingsFile" interface properties. The extension provided for this file is ".SXML".

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
Dim objSymbol As HisLogWndCmdTarget
Set objSymbol = GetSynopticObject.GetSubObject("TestObject").GetObjectInterface
If objSymbol Is Nothing Then Exit Sub
objSymbol.ExtSettingsFile = "test.sxml"
objSymbol.LoadExtSettings
Set objSymbol = Nothing
End Sub
```

RecalcLayout, HisLogWndCmdTarget Function

Syntax RecalcLayout()

Description The function updates the object graphical layout. This function needs to be executed after a property involving the object's graphical aspect, has been edited such as changing the sizes of one of the columns.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
	Dim HisWindow As HisLogWndCmdTarget
	Set HisWindow = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
	If Not HisWindow Is Nothing Then
		HisWindow.AutoLayout = Not HisWindow.AutoLayout
		HisWindow.RecalcLayout
	End If
	Set HisWindow = Nothing
End Sub
```

Refresh, HisLogWndCmdTarget Function

Syntax Refresh()

Description This function refreshes the data in the object which is useful when the query for extracting data from the Log database is edited.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
        Dim HisWindow As HisLogWndCmdTarget
        Set HisWindow = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
        If Not HisWindow Is Nothing Then
                HisWindow.Query = "SELECT * FROM SysMsgs ORDER BY SysMsgs.TimeCol DESC"
                HisWindow.Refresh
        End If
        Set HisWindow = Nothing
End Sub
```

SaveExtSettings, HisLogWndCmdTarget Function

Syntax SaveExtSettings

Description This function permits the objects settings to be save in the relating external settings file. This file can be specified when in design mode in the "Ext. Settings File" property, or using the property from the "ExtSettingsFile" interface. The extension provided for this file is ".SXML".

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
Dim objSymbol As HisLogWndCmdTarget
Set objSymbol = GetSynopticObject.GetSubObject("TestObject").GetObjectInterface
If objSymbol Is Nothing Then Exit Sub
objSymbol.ExtSettingsFile = "test.sxml"
objSymbol.SaveExtSettings
Set objSymbol = Nothing
End Sub
```

AutoLayout, HisLogWndCmdTarget Property

Syntax AutoLayout = _Boolean

Description When enabling this property, the layout will be set to automatic mode. This means that the columns will be automatically resized so that they all fit into the area of the Log Window. When this property is disabled, the columns will show with the sizes set during programming mode when the window is opened. The last columns, on the right, may not fit into the window and will have to be viewed by using the horizontal scroll bar.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
	Dim HisWindow As HisLogWndCmdTarget
	Set HisWindow = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
	If Not HisWindow Is Nothing Then
		HisWindow.AutoLayout = Not HisWindow.AutoLayout
		HisWindow.RecalcLayout
	End If
	Set HisWindow = Nothing
End Sub
```

ButtonPos, HisLogWndCmdTarget Property

Syntax ButtonPos

Description This setting returns the position where the buttons in the Log Window are to appear.

The options are:
0 = left
1 = top
2 = right
3 = botton

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
	Dim objHisLogWnd As HisLogWndCmdTarget
	Set objHisLogWnd = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
	If Not objHisLogWnd Is Nothing Then
		MsgBox "objHisLogWnd's ButtonPos is " & objHisLogWnd.ButtonPos
		,vbInformation,GetProjectTitle
		objHisLogWnd.ButtonPos = 2
		objHisLogWnd.RecalcLayout
	Else
```

```
            MsgBox "objHisLogWnd is nothing",vbInformation,GetProjectTitle
        End If
End Sub
```

ButtonSize, HisLogWndCmdTarget Property

Syntax ButtonSize

Description This setting returns the size of the buttons which are to appear in the Log Window.

The options are:
0 = small
1 = medium
2 = large

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
        Dim objHisLogWnd As HisLogWndCmdTarget
        Set objHisLogWnd = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
        If Not objHisLogWnd Is Nothing Then
                MsgBox "objHisLogWnd's ButtonSize is " & objHisLogWnd.ButtonSize,vbInformation,GetProjectTitle
                objHisLogWnd.ButtonSize= 2
                objHisLogWnd.RecalcLayout
        Else
                MsgBox "objHisLogWnd is nothing",vbInformation,GetProjectTitle
        End If
End Sub
```

Clickable, HisLogWndCmdTarget Property

Syntax Clickable = _Boolean

Description This property is used to define whether the operator can interact with the Historical Log window. When this property is disabled, the control will no longer respond when either clicked by the mouse or operated from keyboard.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
	Dim HisWindow As HisLogWndCmdTarget
	Set HisWindow = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
	If Not HisWindow Is Nothing Then
		HisWindow.Clickable = Not HisWindow.Clickable
		HisWindow.RecalcLayout
	End If
	Set HisWindow = Nothing
End Sub
```

EventType, HisLogWndCmdTarget Property

Syntax EventType = _Integer

Description This property sets or returns the message type to be displayed in the Log Window.

The value options are:

0 = System Messages
1 = Alarm Messages
2 = Comm.Driver Messages
3 = All Messages

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
	Dim HisWindow As HisLogWndCmdTarget
	Set HisWindow = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
	If Not HisWindow Is Nothing Then
		HisWindow.EventType = 1
	End If
	Set HisWindow = Nothing
End Sub
```

ExtSettingsFile, HisLogWndCmdTarget Property

Syntax ExtSettingsFile = _String

Description This property sets or returns the external configuration file for the referenced object. the file can be alsow specified in design mode in the object's "Configuration File" property. The extension provided for this file is ".SXML".

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
Dim objSymbol As HisLogWndCmdTarget
Set objSymbol = GetSynopticObject.GetSubObject("TestObject").GetObjectInterface
If objSymbol Is Nothing Then Exit Sub
objSymbol.ExtSettingsFile = "test.sxml"
objSymbol.SaveExtSettings
Set objSymbol= Nothing
End Sub
```

FilterBtnText, HisLogWndCmdTarget Property

Syntax FilterBtnText = _String

Description This property sets or returns a text for the command button used for printing reports on the data displayed in the Historical Log. When nothing is specified, Movicon will use the default text.

Parameter	Description
None	None

Result String

Example:

```
Sub Click()
        Dim objHisLogWnd As HisLogWndCmdTarget
         Set objHisLogWnd = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
        If Not objHisLogWnd Is Nothing Then
                objHisLogWnd.RefreshBtnText = "Riefresh all"
                objHisLogWnd.PrintBtnText = "Print report"
                objHisLogWnd.FilterBtnText = "Sort by..."
                objHisLogWnd.RecalcLayout
                MsgBox "objHisLogWnd's RefreshBtnText is " & objHisLogWnd.RefreshBtnText
                  ,vbInformation,GetProjectTitle
                MsgBox "objHisLogWnd's PrintBtnText is " & objHisLogWnd.PrintBtnText
                  ,vbInformation,GetProjectTitle
                MsgBox "objHisLogWnd's FilterBtnText is " & objHisLogWnd.FilterBtnText
                  ,vbInformation,GetProjectTitle
        Else
                MsgBox "objHisLogWnd is nothing",vbInformation,GetProjectTitle
        End If
End Sub
```

FilterEvent, HisLogWndCmdTarget Property

Syntax FilterEvent = _String

Description This property sets or returns the Event Type filter for displaying messages in the Movicon Historical Log window.

This filter can have the following values:

ALARM ACK''ALARM OFF
ALARM ON
ALARM RESET
Com. Driver
System
Trace

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
	Dim objHisLogWnd As HisLogWndCmdTarget
	Set objHisLogWnd = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
	If Not objHisLogWnd Is Nothing Then
		MsgBox "objHisLogWnd's FilterEvent is " & objHisLogWnd.FilterEvent
		,vbInformation,GetProjectTitle
		objHisLogWnd.FilterEvent = "System"
		objHisLogWnd.Refresh
	Else
		MsgBox "objHisLogWnd is nothing",vbInformation,GetProjectTitle
	End If
End Sub
```

FilterEventTypeCol, HisLogWndCmdTarget Property

Syntax FilterEventTypeCol = _Long

Description This property sets or returns the 'Property' filter for displaying messages in the Movicon Historical Log window. The priority refers to the Event ID recorded in the EvNumCol. column. This setting only has meaning when a value other than "none" has been entered in the "Priority Condition" field.

Parameter	Description
None	None

Result Long

Example:

```
Option Explicit
Public Sub Click()
	Dim objHisLogWnd As HisLogWndCmdTarget
	Set objHisLogWnd = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
	If Not objHisLogWnd Is Nothing Then
		MsgBox "objHisLogWnd's FilterEventTypeCol is " &
		objHisLogWnd.FilterEventTypeCol,vbInformation,GetProjectTitle
		objHisLogWnd.FilterEventTypeCol = 0
		objHisLogWnd.Refresh
	Else
		MsgBox "objHisLogWnd is nothing",vbInformation,GetProjectTitle
	End If
End Sub
```

FilterEventTypeColCondition, HisLogWndCmdTarget Property

Syntax FilterEventTypeColCondition = _Integer

Description This property sets or returns the 'Severity Condition' filter for displaying messages in the Movicon Historical Log window.

The values which can be used are:

0 -> 'none'
1 -> 'equal'
2 -> 'major'
3 -> 'minor'

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
	Dim objHisLogWnd As HisLogWndCmdTarget
	Set objHisLogWnd = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
	If Not objHisLogWnd Is Nothing Then
		MsgBox "FilterEventTypeColCondition is " &
		objHisLogWnd.FilterEventTypeColCondition,vbInformation,GetProjectTitle
		objHisLogWnd.FilterEventTypeColCondition = 0
		objHisLogWnd.Refresh
	Else
		MsgBox "objHisLogWnd is nothing",vbInformation,GetProjectTitle
	End If
End Sub
```

FilterFromDate, HisLogWndCmdTarget Property

Syntax FilterFromDate = _Date

Description This property sets or returns the 'From Date' filter for displaying messages in the Movicon Historical Log window.

Parameter	Description
None	None

Result Date

Example:

```
Option Explicit
Public Sub Click()
	Dim objHisLogWnd As HisLogWndCmdTarget
	Set objHisLogWnd = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
	If Not objHisLogWnd Is Nothing Then
		MsgBox "objHisLogWnd's FilterFromDate is " & objHisLogWnd.FilterFromDate,vbInformation,GetProjectTitle
		objHisLogWnd.FilterFromDate = Now()
		objHisLogWnd.Refresh
	Else
		MsgBox "objHisLogWnd is nothing",vbInformation,GetProjectTitle
	End If
End Sub
```

FilterToDate, HisLogWndCmdTarget Property

Syntax FilterToDate = _Date

Description This property sets or returns the 'Data finale' filter for displaying messages in the Movicon Historical Log window.

Parameter	Description
None	None

Result Date

Example:

```
Option Explicit
Public Sub Click()
	Dim objHisLogWnd As HisLogWndCmdTarget
	Set objHisLogWnd = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
	If Not objHisLogWnd Is Nothing Then
		MsgBox "objHisLogWnd's FilterToDate is " & objHisLogWnd.FilterToDate,vbInformation,GetProjectTitle
		objHisLogWnd.FilterToDate = Now()
		objHisLogWnd.Refresh
	Else
```

```
                MsgBox "objHisLogWnd is nothing",vbInformation,GetProjectTitle
        End If
End Sub
```

FilterUniqueID, HisLogWndCmdTarget Property

Syntax FilterFromDate = _Long

Description This property allows you to filter the Historical Log window records according to the "UniID" (UniqueID) field value recorded for each alarm in the Alarm table.

Parameter	Description
None	None

Result Long

Example:

```
Option Explicit
Dim objHisLogAlarm As HisLogWndCmdTarget
Dim objAlarm As AlarmCmdTarget
Dim objThreshold As AlarmThresholdCmdTarget

Public Sub Click()
        Set objAlarm = GetAlarm("Alarm VAR00001")
        If Not objAlarm Is Nothing Then
                Set objThreshold = objAlarm.GetAlarmThreshold("DigitalThreshold")
                If Not objThreshold Is Nothing Then
                        Set objHisLogAlarm =
                        GetSynopticObject.GetSubObject("AlarmHisLog").GetObjectInterf
                        ace
                        If Not objHisLogAlarm Is Nothing Then
                                objHisLogAlarm.FilterUniqueID =
                                objThreshold.GetUniqueID
                                objHisLogAlarm.Refresh
                        End If
                End If
        End If
        Set objAlarm = Nothing
        Set objThreshold = Nothing
        Set objHisLogAlarm = Nothing
End Sub
```

FilterUser, HisLogWndCmdTarget Property

Syntax FilterUser = _String

Description This property sets or returns the 'Utente' filter for displaying messages in the Movicon Historical Log window.

Parameter	Description

None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
	Dim objHisLogWnd As HisLogWndCmdTarget
	Set objHisLogWnd = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
	If Not objHisLogWnd Is Nothing Then
		MsgBox "objHisLogWnd's FilterUser is " & objHisLogWnd.FilterUser,vbInformation,GetProjectTitle
		objHisLogWnd.FilterUser = "User00001"
		objHisLogWnd.Refresh
	Else
		MsgBox "objHisLogWnd is nothing",vbInformation,GetProjectTitle
	End If
End Sub
```

FormatDateTime, HisLogWndCmdTarget Property

Syntax FormatDateTime = _String

Description This property allows you to insert the date and time format to be used for displaying the time in the Historical Log's "Event Time" column. All the format codes that can be used in this property are listed in the Drawings and Controls Proprietà Stile section. After having modified this property, you must "Refresh" Historical Log window.

Parameter	Description
None	None

Result String

Example:

```
Dim WndHisLog As HisLogWndCmdTarget
Public Sub Click()
	Set WndHisLog =GetSynopticObject.GetSubObject("objHisLog").GetObjectInterface
	WndHisLog.FormatDateTime ="%d:%m:%Y %H:%M:%S"
	WndHisLog.Refresh
	Set WndHisLog = Nothing
End Sub
```

FormatDuration, HisLogWndCmdTarget Property

Syntax FormatDuration = _String

Description This property allows you to insert the format of the duration shown in the Historical Log window's "Duration" column. All the format codes that can be used in this property are listed in the Drawings and Controls Stile Properties section. After having modified this property you will need to "Refresh" the Historical Log window.

Parameter	Description
None	None

Result String

Example:

```
Dim WndHisLog As HisLogWndCmdTarget
Public Sub Click()
        Set WndHisLog =
        GetSynopticObject.GetSubObject("objHisLog").GetObjectInterface
        WndHisLog.FormatDuration="%D,%H:%M:%S"
        WndHisLog.Refresh
        Set WndHisLog=Nothing
End Sub
```

GraphicButtons, HisLogWndCmdTarget Property

Syntax GraphicButtons = _Boolean

Description When Enabling this property, the Historical Log Window buttons are drawn using an icon instead of text. The text will instead be displayed as a tooltip when positioning the mouse on top of the button.

The tooltip is not managed in Windows CE versions.

This property is not managed by the 'Alarm Banner' object.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Click()
        GraphicButtons = True
        RecalcLayout
End Sub
```

NetworkBackupServerName, HisLogWndCmdTarget Property

Syntax NetworkBackupServerName = _String

Description This property sets or returns the name of any Network Backup Server used for getting data to display in the Historical Log window when the primary server, the one set in the 'NetowrkServerName'property is in timeout.

Parameter	Description
None	None

Result String

Example:

```
Dim objHislLogWnd As HisLogWndCmdTarget
Public Sub Click()
        Debug.Print objHislLogWnd.NetworkBackupServerName
End Sub
Public Sub SymbolLoading()
        Set objHislLogWnd=
        GetSynopticObject.GetSubObject("HisLogWindow").GetObjectInterface
End Sub
```

IncludeMilliseconds, HisLogWndCmdTarget Property

Syntax IncludeMilliseconds = _Boolean

Description This property permits you to define whether to display or not to display the milliseconds in the window's time column.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
        Dim HisWindow As HisLogWndCmdTarget
        Set HisWindow = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
        If Not HisWindow Is Nothing Then
                HisWindow.IncludeMilliseconds = Not HisWindow.IncludeMilliseconds
                HisWindow.Refresh
        End If
        Set HisWindow = Nothing
End Sub
```

MaxCount, HisLogWndCmdTarget Property

Syntax MaxCount = _Integer

Description This property allows you set the maximum number of rows to be displayed in the Log Window.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
        Dim HisWindow As HisLogWndCmdTarget
        Set HisWindow = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
        If Not HisWindow Is Nothing Then
                HisWindow.MaxCount = 150
                HisWindow.Refresh
        End If
        Set HisWindow = Nothing
End Sub
```

NetworkBackupServerName, HisLogWndCmdTarget Property

Syntax NetworkBackupServerName = _String

Description This property sets or returns the name of any Network Backup Server used for getting data to display in the Historical Log window when the primary server, the one set in the 'NetowrkServerName'property is in timeout.

Parameter	Description
None	None

Result String

Example:

```
Dim objHislLogWnd As HisLogWndCmdTarget
Public Sub Click()
        Debug.Print objHislLogWnd.NetworkBackupServerName
End Sub
Public Sub SymbolLoading()
        Set objHislLogWnd=
        GetSynopticObject.GetSubObject("HisLogWindow").GetObjectInterface
```

```
End Sub
```

NetworkServerName, HisLogWndCmdTarget Property

Syntax NetworkServerName = _String

Description This property returns the name of any Network Server where data is to be retrieved for displaying in the Historical Log window.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim HisWindow As HisLogWndCmdTarget
        Set HisWindow = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
        If Not HisWindow Is Nothing Then
                HisWindow.NetworkServerName = "PERSONAL11"
                HisWindow.Refresh
        End If
        Set HisWindow = Nothing
End Sub
```

PrintBtnText, HisLogWndCmdTarget Property

Syntax PrintBtnText = _String

Description This property sets or returns a text for the command button for printing a report on the data displayed in the Historical Log window. When nothing is specified, Movicon will use the default text.

This property is not supported in Windows CE.(If set, always returns an empty string)

Parameter	Description
None	None

Result String

Example:

```
Sub Click()
        Dim objHisLogWnd As HisLogWndCmdTarget
```

```
 Set objHisLogWnd = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
If Not objHisLogWnd Is Nothing Then
        objHisLogWnd.RefreshBtnText = "Riefresh all"
        objHisLogWnd.PrintBtnText = "Print report"
        objHisLogWnd.FilterBtnText = "Sort by..."
        objHisLogWnd.RecalcLayout
        MsgBox "objHisLogWnd's RefreshBtnText is " & objHisLogWnd.RefreshBtnText
          ,vbInformation,GetProjectTitle
        MsgBox "objHisLogWnd's PrintBtnText is " & objHisLogWnd.PrintBtnText
          ,vbInformation,GetProjectTitle
        MsgBox "objHisLogWnd's FilterBtnText is " & objHisLogWnd.FilterBtnText
          ,vbInformation,GetProjectTitle
Else
        MsgBox "objHisLogWnd is nothing",vbInformation,GetProjectTitle
End If
End Sub
```

Project, HisLogWndCmdTarget Property

Syntax Project = _String

Description This property allows you to set the name of the child project from which you wish to recuperate data to be displayed. The current project will be used if this field is left blank.

The name of the eventual child project of the current project is to be inserted exclusively.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim HisWindow As HisLogWndCmdTarget
        Set HisWindow = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
        If Not HisWindow Is Nothing Then
                Debug.Print HisWindow.Project
        End If
        Set HisWindow = Nothing
End Sub
```

Query, HisLogWndCmdTarget Property

Syntax Query = _String

Description This property allows you to set a selection Query in SQL language for extracting data contained in the Log database. This query is executed for default upon each data refresh in the window, whether executed automatically or on the operator's command.

There are three tables in the Log database:

- SysMsgs System Messages
- Drivers Messages relating to Drivers
- Alarms Messages relating to Alarms

The columns which are used to structure the tables are:

- EventCol Event ID
- EvDescCol Event Text
- TimeCol Event Time
- UserCol User

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
	Dim HisWindow As HisLogWndCmdTarget
	Set HisWindow = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
	If Not HisWindow Is Nothing Then
		HisWindow.Query = "SELECT * FROM SysMsgs ORDER BY SysMsgs.TimeCol DESC"
		HisWindow.Refresh
	End If
	Set HisWindow = Nothing
End Sub
```

RefreshBtnText, HisLogWndCmdTarget Property

Syntax RefreshBtnText = _String

Description This property sets or returns a text for the command button which refreshes data displayed in the Historical Log window. When nothing is specified, Movicon will use the default text.

Parameter	Description
None	None

Result String

Example:

```
Sub Click()
	Dim objHisLogWnd As HisLogWndCmdTarget
	 Set objHisLogWnd = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
	If Not objHisLogWnd Is Nothing Then
		objHisLogWnd.RefreshBtnText = "Riefresh all"
		objHisLogWnd.PrintBtnText = "Print report"
		objHisLogWnd.FilterBtnText = "Sort by..."
		objHisLogWnd.RecalcLayout
```

```
            MsgBox "objHisLogWnd's RefreshBtnText is " & objHisLogWnd.RefreshBtnText
              ,vbInformation,GetProjectTitle
            MsgBox "objHisLogWnd's PrintBtnText is " & objHisLogWnd.PrintBtnText
              ,vbInformation,GetProjectTitle
            MsgBox "objHisLogWnd's FilterBtnText is " & objHisLogWnd.FilterBtnText
              ,vbInformation,GetProjectTitle
        Else
            MsgBox "objHisLogWnd is nothing",vbInformation,GetProjectTitle
        End If
End Sub
```

ReportFile, HisLogWndCmdTarget Property

Syntax ReportFile = _String

Description This property sets or returns the name of the report file to be used for printing data displayed in the Historical Log window. The file must be created with the Report Designer or Crystal Report© (.rpt). If this field is left empty, Movivon will use the default report file created by Progea in the Report Designer format.

This property is not supported in Windows CE.(when used, always returns an empty string)

Parameter	Description
None	None

Result String

Example:

```
Sub Click()
        Dim objHisLogWnd As HisLogWndCmdTarget
         Set objHisLogWnd = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
        If Not objHisLogWnd Is Nothing Then
            objHisLogWnd.ReportFile = "C:\Report1.rpt"
        Else
            MsgBox "objHisLogWnd is nothing",vbInformation,GetProjectTitle
        End If
End Sub
```

ShowFilterBtn, HisLogWndCmdTarget Property

Syntax ShowFilterBtn = _Boolean

Description This property allows you to display the command button for filtering data in the Historical Log window.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
	Dim HisWindow As HisLogWndCmdTarget
	Set HisWindow = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
	If Not HisWindow Is Nothing Then
		HisWindow.ShowFilterBtn = Not HisWindow.ShowFilterBtn
		HisWindow.RecalcLayout
	End If
	Set HisWindow = Nothing
End Sub
```

ShowPrintBtn, HisLogWndCmdTarget Property

Syntax ShowPrintBtn = _Boolean

Description This property allows the command button to be shown for printing data from the Historical Log window.
The print is executed using the report fiel which should be specified in the "Report File" property. Movicon passes the same filter settings to the report for printing data which coincide with the data shown in the window in question.

This property is not suupported in Windows CE.(If set, always returns 'false')

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
	Dim HisWindow As HisLogWndCmdTarget
	Set HisWindow = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
	If Not HisWindow Is Nothing Then
		HisWindow.ShowPrintBtn = Not HisWindow.ShowPrintBtn
		HisWindow.RecalcLayout
	End If
	Set HisWindow = Nothing
End Sub
```

ShowRefreshBtn, HisLogWndCmdTarget Property

Syntax ShowRefreshBtn = _Boolean

Description This property allows you to show the command button for refreshing data in the Historical Log display window.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
        Dim HisWindow As HisLogWndCmdTarget
        Set HisWindow = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
        If Not HisWindow Is Nothing Then
                HisWindow.ShowRefreshBtn = Not HisWindow.ShowRefreshBtn
                HisWindow.RecalcLayout
        End If
        Set HisWindow = Nothing
End Sub
```

SortBy, HisLogWndCmdTarget Property

Syntax SortBy = _String

Description This property sets or returns the 'Sort By' filter for displaying messages in the Movicon Historical Log window.

The possible fields are:

CommCol
DescCol
DurCol
EvDescCol
EvCol
EvNumCol
LocalCol
MSecCol
TimeCol
UserCol

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim objHisLogWnd As HisLogWndCmdTarget
        Set objHisLogWnd = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
        If Not objHisLogWnd Is Nothing Then
```

```
            MsgBox "objHisLogWnd's SortBy is " & objHisLogWnd.SortBy
            ,vbInformation,GetProjectTitle
            objHisLogWnd.SortBy = "EvCol"
            objHisLogWnd.Refresh
        Else
            MsgBox "objHisLogWnd is nothing",vbInformation,GetProjectTitle
        End If
End Sub
```

SubItemComment, HisLogWndCmdTarget Property

Syntax SubItemComment = _String

Description This property allows you to set the text which is to appear as the "Comment" column's name. The default text will be used if this field is left blank.

Parameter	Description
None	None

Result String

Example:

```
Dim objHisLogWnd As HisLogWndCmdTarget
Public Sub Click()
    Debug.Print objHisLogWnd.SubItemEventNum
End Sub
Public Sub SymbolLoading()
    Set objHisLogWnd = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
End Sub
```

SubItemCommentPos, HisLogWndCmdTarget Property

Syntax SubItemCommentPos = _Integer

Description This property sets or returns the position of the "Comment" column within Log Window. When setting a new value, the other columns will be automatically re-positioned in the window layout. In addition when setting the "-1", the column will be hidden. The "0" value is used to indicate position of the first column on the left in the window.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
	Dim HisWindow As HisLogWndCmdTarget
	Set HisWindow = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
	If Not HisWindow Is Nothing Then
		Debug.Print HisWindow.SubItemCommentPos
	End If
	Set HisWindow = Nothing
End Sub
```

SubItemCommentWidth, HisLogWndCmdTarget Property

Syntax SubItemCommentWidth = _Integer

Description This property indicates the column's size in pixels in the Historical Log window. If this column is not displayed, the -1 value will be returned.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
	Dim HisWindow As HisLogWndCmdTarget
	Set HisWindow = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
	If Not HisWindow Is Nothing Then
		HisWindow.SubItemCommentWidth = 20
		HisWindow.RecalcLayout
	End If
	Set HisWindow = Nothing
End Sub
```

SubItemDesc, HisLogWndCmdTarget Property

Syntax SubItemDesc = _String

Description This property allows you to set the text which is to appear as the "Description" column's name. When this field is left blank, the default text will be used instead.

Parameter	Description
None	None

Result String

Example:

```
Dim objHisLogWnd As HisLogWndCmdTarget
Public Sub Click()
	Debug.Print objHisLogWnd.SubItemDesc
End Sub
Public Sub SymbolLoading()
	Set objHisLogWnd = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
End Sub
```

SubItemDescPos, HisLogWndCmdTarget Property

Syntax SubItemDescPos = _Integer

Description This property sets or returns the position of the "Description" column within Log Window. When setting a new value, the other columns will be automatically re-positioned in the window layout. In addition when setting the "-1", the column will be hidden. The "0" value is used to indicate position of the first column on the left in the window.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
	Dim HisWindow As HisLogWndCmdTarget
	Set HisWindow = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
	If Not HisWindow Is Nothing Then
		Debug.Print HisWindow.SubItemDescPos
	End If
	Set HisWindow = Nothing
End Sub
```

SubItemDescWidth, HisLogWndCmdTarget Property

Syntax SubItemDescWidth = _Integer

Description This property indicates the size in pixels of the column in the Historical Log display window. When this column is not displayed the -1 value will be returned.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
        Dim HisWindow As HisLogWndCmdTarget
        Set HisWindow = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
        If Not HisWindow Is Nothing Then
                HisWindow.SubItemDescWidth = 20
                HisWindow.RecalcLayout
        End If
        Set HisWindow = Nothing
End Sub
```

SubItemDuration, HisLogWndCmdTarget Property

Syntax SubItemDuration = _String

Description This property allows you to set the text which is to appear as the "Duration" column's name. The default text will be used if this field is left blank.

Parameter	Description
None	None

Result String

Example:

```
Dim objHisLogWnd As HisLogWndCmdTarget
Public Sub Click()
        Debug.Print objHisLogWnd.SubItemDuration
End Sub
Public Sub SymbolLoading()
        Set objHisLogWnd = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
End Sub
```

SubItemDurationPos, HisLogWndCmdTarget Property

Syntax SubItemDurationPos = _Integer

Description This property sets or returns the position of the "Duration" column within Historical Log window. When setting a new value, the other columns will be automatically re-positioned in the window layout. In addition when setting the "-1", the column will be hidden. The "0" value is used to indicate position of the first column on the left in the window.

Parameter	Description

None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
	Dim HisWindow As HisLogWndCmdTarget
	Set HisWindow = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
	If Not HisWindow Is Nothing Then
		Debug.Print HisWindow.SubItemDurationPos
	End If
	Set HisWindow = Nothing
End Sub
```

SubItemDurationWidth, HisLogWndCmdTarget Property

Syntax SubItemDurationWidth = _Integer

Description This property indicates the size in pixels of the column in the Historical Log display window. When this column is not displayed the -1 value will be returned.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
	Dim HisWindow As HisLogWndCmdTarget
	Set HisWindow = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
	If Not HisWindow Is Nothing Then
		HisWindow.SubItemDurationWidth = 20
		HisWindow.RecalcLayout
	End If
	Set HisWindow = Nothing
End Sub
```

SubItemEventId, HisLogWndCmdTarget Property

Syntax SubItemEventId = _String

Description Allows you to set the text which is to appear as the "Event Id" column's name. The default text will be used if this field is left blank.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim HisWindow As HisLogWndCmdTarget
        Set HisWindow = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
        If Not HisWindow Is Nothing Then
                HisWindow.SubItemEventId = "Message Type"
                HisWindow.RecalcLayout
        End If
        Set HisWindow = Nothing
End Sub
```

SubItemEventIdPos, HisLogWndCmdTarget Property

Syntax SubItemEventIdPos = _String

Description This property sets or returns the position of the "Event ID" column within Historical Log window. When setting a new value, the other columns will be automatically re-positioned in the window layout. In addition when setting the "-1", the column will be hidden. The "0" value is used to indicate position of the first column on the left in the window.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim HisWindow As HisLogWndCmdTarget
        Set HisWindow = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
        If Not HisWindow Is Nothing Then
                Debug.Print HisWindow.SubItemEventIdPos
        End If
        Set HisWindow = Nothing
End Sub
```

SubItemEventIdWidth, HisLogWndCmdTarget Property

Syntax SubItemEventIdWidth = _String

Description This property indicates the size in pixels of the column in the Historical Log display window. The -1 value is returned when the column is not displayed.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
	Dim HisWindow As HisLogWndCmdTarget
	Set HisWindow = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
	If Not HisWindow Is Nothing Then
		HisWindow.SubItemEventIdWidth = 20
		HisWindow.RecalcLayout
	End If
	Set HisWindow = Nothing
End Sub
```

SubItemEventNum, HisLogWndCmdTarget Property

Syntax SubItemEventNum = _String

Description This property allows you to set the text which is to be used as the "Event Number" column. If this field is left blank, the default text will be used instead.

Parameter	Description
None	None

Result String

Example:

```
Dim objHisLogWnd As HisLogWndCmdTarget
Public Sub Click()
	Debug.Print objHisLogWnd.SubItemEventNum
End Sub
Public Sub SymbolLoading()
	Set objHisLogWnd = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
End Sub
```

SubItemEventNumPos, HisLogWndCmdTarget Property

Syntax SubItemEventNumPos = _Integer

Description This property sets or returns the position of the "Event Number ID" column within Historical Log window. When setting a new value, the other columns will be automatically re-positioned in the window layout. In addition when setting the "-1", the column will be hidden. The "0" value is used to indicate position of the first column on the left in the window.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
	Dim HisWindow As HisLogWndCmdTarget
	Set HisWindow = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
	If Not HisWindow Is Nothing Then
		Debug.Print HisWindow.SubItemEventNumPos
	End If
	Set HisWindow = Nothing
End Sub
```

SubItemEventNumWidth, HisLogWndCmdTarget Property

Syntax SubItemEventNumWidth = _Integer

Description This property indicated the size in pixels of the column in the Historical Log display window. If this column is not displayed the -1 value will be returned.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
	Dim HisWindow As HisLogWndCmdTarget
	Set HisWindow = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
	If Not HisWindow Is Nothing Then
		HisWindow.SubItemEventNumWidth = 20
		HisWindow.RecalcLayout
	End If
	Set HisWindow = Nothing
```

```
End Sub
```

SubItemText, HisLogWndCmdTarget Property

Syntax SubItemText = _String

Description Allows you to set the text which is to appear as the "Event Text" column's name. The default text will be used if this field is left blank.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
	Dim HisWindow As HisLogWndCmdTarget
	Set HisWindow = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
	If Not HisWindow Is Nothing Then
		HisWindow.SubItemText = "Description"
		HisWindow.RecalcLayout
	End If
	Set HisWindow = Nothing
End Sub
```

SubItemTextPos, HisLogWndCmdTarget Property

Syntax SubItemTextPos = _Integer

Description This property sets or returns the position of the "Text" column within the Hstorical Log window. When setting a new value, the other columns will be automatically re-positioned in the window layout. In addition when setting the "-1", the column will be hidden. The "0" value is used to indicate position of the first column on the left in the window.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
	Dim HisWindow As HisLogWndCmdTarget
	Set HisWindow = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
	If Not HisWindow Is Nothing Then
```

```
            Debug.Print HisWindow.SubItemTextPos
        End If
        Set HisWindow = Nothing
End Sub
```

SubItemTextWidth, HisLogWndCmdTarget Property

Syntax SubItemTextWidth = _Integer

Description This property indicates the size in pixels of the column in the Historical Log display window. The -1 value is returned when the column is not displayed.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
        Dim HisWindow As HisLogWndCmdTarget
        Set HisWindow = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
        If Not HisWindow Is Nothing Then
            HisWindow.SubItemTextWidth = 20
            HisWindow.RecalcLayout
        End If
        Set HisWindow = Nothing
End Sub
```

SubItemTime, HisLogWndCmdTarget Property

Syntax SubItemTime = _String

Description Allows you to set the text which is to appear as the "Time" column's name. The default text will be used if this field is left blank.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim HisWindow As HisLogWndCmdTarget
        Set HisWindow = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
        If Not HisWindow Is Nothing Then
```

```
            HisWindow.SubItemTime = "Date/Time"
            HisWindow.RecalcLayout
        End If
        Set HisWindow = Nothing
End Sub
```

SubItemTimePos, HisLogWndCmdTarget Property

Syntax SubItemTimePos = _Integer

Description This property sets or returns the position of the "Time" column within the Hstorical Log window. When setting a new value, the other columns will be automatically re-positioned in the window layout. In addition when setting the "-1", the column will be hidden. The "0" value is used to indicate position of the first column on the left in the window.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
        Dim HisWindow As HisLogWndCmdTarget
        Set HisWindow = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
        If Not HisWindow Is Nothing Then
            Debug.Print HisWindow.SubItemTimePos
        End If
        Set HisWindow = Nothing
End Sub
```

SubItemTimeWidth, HisLogWndCmdTarget Property

Syntax SubItemTimeWidth = _Integer

Description This property indicates the size in pixels of the column in the Historical Log display window. The -1 value is returned when the column is not displayed.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
        Dim HisWindow As HisLogWndCmdTarget
        Set HisWindow = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
        If Not HisWindow Is Nothing Then
                HisWindow.SubItemTimeWidth = 20
                HisWindow.RecalcLayout
        End If
        Set HisWindow = Nothing
End Sub
```

SubItemUser, HisLogWndCmdTarget Property

Syntax SubItemUser = _String

Description Allows you to set the text which is to appear as the "User" column's name. The default text will be used if this field is left blank.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim HisWindow As HisLogWndCmdTarget
        Set HisWindow = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
        If Not HisWindow Is Nothing Then
                HisWindow.SubItemUser = "Logon Users"
                HisWindow.RecalcLayout
        End If
        Set HisWindow = Nothing
End Sub
```

SubItemUserPos, HisLogWndCmdTarget Property

Syntax SubItemUserPos = _Integer

Description This property sets or returns the position of the "User" column within the Hstorical Log window. When setting a new value, the other columns will be automatically re-positioned in the window layout. In addition when setting the "-1", the column will be hidden. The "0" value is used to indicate position of the first column on the left in the window.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
	Dim HisWindow As HisLogWndCmdTarget
	Set HisWindow = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
	If Not HisWindow Is Nothing Then
		Debug.Print HisWindow.SubItemUserPos
	End If
	Set HisWindow = Nothing
End Sub
```

SubItemUserWidth, HisLogWndCmdTarget Property

Syntax SubItemUserWidth = _Integer

Description This property indicates the size in pixels of the column in the Historical Log display window. The -1 value is returned when the column is not displayed.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
	Dim HisWindow As HisLogWndCmdTarget
	Set HisWindow = GetSynopticObject.GetSubObject("HisLog").GetObjectInterface
	If Not HisWindow Is Nothing Then
		HisWindow.SubItemUserWidth = 20
		HisWindow.RecalcLayout
	End If
	Set HisWindow = Nothing
End Sub
```

35. HourSelectorCmdTarget

OnAddScheduler, HourSelectorCmdTarget Event

Description Event notified on command used for adding new schedulers in runtime.

Parameter	Description
bRet As Boolean	Set at "false" consents operation annulment.

Example:

```
Public Sub OnAddScheduler(ByRef bRet As Boolean)
        If MsgBox ("Do you want add a new scheduler objects ?", vbYesNo +
        vbQuestion, GetProjectTitle) = vbYes Then
                Debug.Print "Adding Scheduler..."
        Else
                Debug.Print "Deleting Add Scheduler..."
                bRet = False
        End If
End Sub
```

OnCancel, HourSelectorCmdTarget Event

Description Event notified each time a request is made to cancel changes made to the Hour Selector's data.

Parameter	Description
bRet As Boolean	Enabled on status change.

OnRemoveScheduler, HourSelectorCmdTarget Event

Description Event notified on command used for removing a scheduler added in runtime.

Parameter	Description
bRet As Boolean	Set at "false" consents operation cancellation.

Example:

```
Public Sub OnAddScheduler(ByRef bRet As Boolean)
        If MsgBox ("Do you want remove the scheduler objects?", vbYesNo +
        vbQuestion, GetProjectTitle) = vbYes Then
                Debug.Print "Removing Scheduler..."
        Else
                Debug.Print "Deleting Remove Scheduler..."
                bRet = False
        End If
End Sub
```

OnSave, HourSelectorCmdTarget Event

Description Event notified each time a request is made to save changes made to the Scheduler's Hour Selector.

Parameter	Description
bRet As Boolean	Enabled on status change.

OnSchedulerChanged, HourSelectorCmdTarget Event

Description Event notified every time the scheduler object associated to the window changes value, for instance: when another is selected using the scheduler selection combo box.

Parameter	Description
bRet As Boolean	

Example:

```
Public Sub OnSchedulerChanged()
        Dim objScheduler As SchedulerCmdTarget
        If Scheduler = "" Then Exit Sub
        Set objScheduler = GetScheduler(Scheduler)
        If Not objScheduler Is Nothing Then
                MsgBox "New Scheduler Name: " & objScheduler.Name
        End If
        Set objScheduler = Nothing
End Sub
```

OnSwitchGridMode, HourSelectorCmdTarget Event

Description Event notifies on command used for passing form table mode to grid mode and viceversa.

Parameter	Description
bRet As Boolean	Set at "false" consents operation cancellation.

Example:

```
Public Sub OnAddScheduler(ByRef bRet As Boolean)
        If MsgBox ("Do you want switch to grid mode ?", vbYesNo + vbQuestion,
        GetProjectTitle) = vbYes Then
                Debug.Print "Swtiching to grid mode..."
        Else
                Debug.Print "Deleting Swtich to grid mode..."
                bRet = False
        End If
End Sub
```

OnSwitchHolidays, HourSelectorCmdTarget Event

Description Event notifies on command used for passing from normal hour plan to holiday plan and viceversa.

Parameter	Description
bRet As Boolean	Set at "false" consents operation cancellation.

Example:

```
Public Sub OnAddScheduler(ByRef bRet As Boolean)
        If MsgBox ("Do you want switch to holiday ?", vbYesNo + vbQuestion,
        GetProjectTitle) = vbYes Then
                Debug.Print "Swtiching to holiday..."
        Else
                Debug.Print "Deleting Swtich to holiday..."
                bRet = False
        End If
End Sub
```

Cancel, HourSelectorCmdTarget Function

Syntax Cancel()

Description This function cancels every change made to the hour selector's data and reloads the plan from the scheduler's file. The property returns True when this operation is successful. This method can also be used for refreshing the scheduler after changes have been made to the planning directly on file.

Parameter	Description
None	None

Result Boolean

Example:

```
Dim objSelector As HourSelectorCmdTarget
Public Sub Click()
        Debug.Print objSelector.Cancel
End Sub
Public Sub SymbolLoading()
        Set objSelector =
        GetSynopticObject.GetSubObject("HourSelector").GetObjectInterface
End Sub
```

LoadExtSettings, HourSelectorCmdTarget Function

Syntax LoadExtSettings

Description This function permits the object's relating external file settings to be loaded. This file can be specified in design mode in the "External File settings" property or in the "ExtSettingsFile" interface properties. The extension provided for this file is ".SXML".

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
Dim objSymbol As HourSelectorWndCmdTarget
Set objSymbol = GetSynopticObject.GetSubObject("TestObject").GetObjectInterface
If objSymbol Is Nothing Then Exit Sub
objSymbol.ExtSettingsFile = "test.sxml"
objSymbol.LoadExtSettings
Set objSymbol = Nothing
End Sub
```

RecalcLayout, HourSelectorCmdTarget Function

Syntax RecalcLayout()

Description This function recalculates the object's layout. This function needs to be executes after change has been made any property relating to the object's layout.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Click()
        Dim bResult As Boolean
        bResult = RecalcLayout
        Debug.Print bResult
End Sub
```

Save, HourSelectorCmdTarget Function

Syntax Save()

Description This function saves each change made to the Hour Selector's data and returns True when save has been completed successfully.

Parameter	Description
None	None

Result Boolean

Example:

```
Dim objSelector As HourSelectorCmdTarget
Public Sub Click()
        Debug.Print objSelector.Save
End Sub
Public Sub SymbolLoading()
        Set objSelector =
        GetSynopticObject.GetSubObject("HourSelector").GetObjectInterface
End Sub
```

SaveExtSettings, HourSelectorCmdTarget Function

Syntax SaveExtSettings

Description This function permits the objects settings to be save in the relating external settings file. This file can be specified when in design mode in the "Ext. Settings File" property, or using the property from the "ExtSettingsFile" interface. The extension provided for this file is ".SXML".

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
Dim objSymbol As HourSelectorWndCmdTarget
Set objSymbol = GetSynopticObject.GetSubObject("TestObject").GetObjectInterface
If objSymbol Is Nothing Then Exit Sub
objSymbol.ExtSettingsFile = "test.sxml"
objSymbol.SaveExtSettings
Set objSymbol = Nothing
End Sub
```

AddBtnText, HourSelectorCmdTarget Property

Syntax AddBtnText = _String

Description This property sets or returns the test displayed in the button used for adding new schedulers in runtime. Setting this property with an empty string will display the predefined text. Edits will only be effective after having called the "RecalcLayout" method.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim objHourSelector As HourSelectorCmdTarget

        If objHourSelector Is Nothing Then Set objHourSelector =
        GetSynopticObject.GetSubObject("HourSelector1").GetObjectInterface
        If objHourSelector Is Nothing Then Exit Sub
        If objHourSelector.AddBtnText = "" Then
                objHourSelector.AddBtnText = "ADD (INS)"
        Else
                objHourSelector.AddBtnText = ""
        End If
        objHourSelector.RecalcLayout
End Sub
```

Border, HourSelectorCmdTarget Property

Syntax Border = _Integer

Description This property sets or returns the border type set for the object according to the proposed None, bump, etched, raised or sunken options which are also available in the object's general properties.
The following values are validi: 0=none, 1=bump, 2=etched, 3=raised, 4=sunken.

Parameter	Description

None	None

Result Integer

Example:

```
Public Sub Click()
        For i = 0 To 4 Step 1
        Border= i
        sRet = Border
        MsgBox "Border= " & sRet, vbOkOnly, GetProjectTitle
        Next i
End Sub
```

ButtonPos, HourSelectorCmdTarget Property

Syntax ButtonPos = _Integer

Description This property sets or returns the position for the "Save" and "Cancel" buttons according to the proposed Left, Top, Right, Bottom options which are also available in the object's general properties.
The following values are valid: 0=Left, 1=Top, 2=Right, 3=Bottom.

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        For i = 0 To 3 Step 1
        ButtonPos= i
        sRet = ButtonPos
        MsgBox "ButtonPos= " & sRet, vbOkOnly, GetProjectTitle
        Next i
End Sub
```

ButtonSize, HourSelectorCmdTarget Property

Syntax ButtonSize = _Integer

Description This property sets or returns the "Save" and "Cancel" button sizes according to the Small, Medium and large options which are also available in the object's style properties.
The following values are valid: 0=Small, 1=Medium, 2=Large.

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        For i = 0 To 2 Step 1
        ButtonSize= i
        sRet = ButtonSize
        MsgBox "ButtonSize= " & sRet, vbOkOnly, GetProjectTitle
        Next i
End Sub
```

CancelBtnText, HourSelectorCmdTarget Property

Syntax CancelBtnText = _String

Description This property sets or returns a text for the Cancel command button used for cancelling the Hour Selector's data modifications. The default text will be used if nothing has been entered.

Parameter	Description
None	None

Result String

Example:

```
Dim objSelector As HourSelectorCmdTarget
Public Sub Click()
        Debug.Print objSelector.CancelBtnText
End Sub
Public Sub SymbolLoading()
        Set objSelector =
        GetSynopticObject.GetSubObject("HourSelector").GetObjectInterface
        objSelector.CancelBtnText = "Cancella"
End Sub
```

ColorSelCell, HourSelectorCmdTarget Property

Syntax ColorSelCell = _Long

Description This property sets or returns the colour selected for the Hour Selector Window.

Parameter	Description
None	None

Result Long

Example:

```
Dim objSelector As HourSelectorCmdTarget
Public Sub Click()
        Debug.Print objSelector.ColorSelCell
End Sub
Public Sub SymbolLoading()
        Set objSelector =
        GetSynopticObject.GetSubObject("HourSelector").GetObjectInterface
End Sub
```

DaysText, HourSelectorCmdTarget Property

Syntax DaysText(_nIndex) = _String

Description This property sets or returns the text displayed for the horizontal boxes on the left hand side. The index can have values starting from 0 to 7. When left blank, Movicon will use the default text.

Parameter	Description
nIndex As Integer	Index of reference box

Result Long

Example:

```
Dim objSelector As HourSelectorCmdTarget
Public Sub Click()
        Debug.Print objSelector.DaysText(1)
End Sub
Public Sub SymbolLoading()
        Set objSelector =
        GetSynopticObject.GetSubObject("HourSelector").GetObjectInterface
        objSelector.DaysText(1) = "Domenica"
End Sub
```

EditMode, HourSelectorCmdTarget Property

Syntax EditMode = _Integer

Description This property sets or returns the mode with which hour plan input is executed. Changes are put into effect only after the "RecalcLayout" method has been called. The following values are valid:

0 = Time Table: standard display, with the option to set activation/deactivation timeframes with 15 min. precisions.
1 = Grid: Grid display with option to set activation/deactivation timeframes with minute precision.
2 = Both: A button activates for passing back and forth between the Grid display (preset as opening display), and the TimeTable display.

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        Dim objHourSelector As HourSelectorCmdTarget
        If objHourSelector Is Nothing Then Set objHourSelector =
        GetSynopticObject.GetSubObject("HourSelector1").GetObjectInterface
        If objHourSelector Is Nothing Then Exit Sub

        Dim sEditMode(0 To 2) As String
        Dim nSelectItem As Integer

        sEditMode(0) = "Time Table"
        sEditMode(1) = "Grid"
        sEditMode(2) = "Both"
        nSelectItem = ShowPopupMenu(sEditMode, , , )
        objHourSelector.EditMode = nSelectItem
        objHourSelector.RecalcLayout
End Sub
```

EndTimeColText, HourSelectorCmdTarget Property

Syntax EndTimeColText = _String

Description This property sets or returns the text displayed in the grid's second column's title. Setting this property with an empty string will display the predefined text. Changes will take effect only after the "RecalcLayout" method has been called.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim objHourSelector As HourSelectorCmdTarget
        If objHourSelector Is Nothing Then Set objHourSelector =
        GetSynopticObject.GetSubObject("HourSelector1").GetObjectInterface
        If objHourSelector Is Nothing Then Exit Sub
        If objHourSelector.EndTimeColText = "" Then
                objHourSelector.EndTimeColText = "END"
        Else
                objHourSelector.EndTimeColText = ""
        End If
        objHourSelector.RecalcLayout
End Sub
```

ErrorString, HourSelectorCmdTarget Property

Syntax ErrorString = _String

Description This property sets or returns the text which will be displayed as the error sting during the hour plan setting. If the user inserts an incorrect timeframe while editing in Grid mode, the text corresponding to the this property's value will be shown as the error message's text. When setting this property with an empty string, the predefined text will be displayed.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim objHourSelector As HourSelectorCmdTarget
        If objHourSelector Is Nothing Then Set objHourSelector =
        GetSynopticObject.GetSubObject("HourSelector1").GetObjectInterface
        If objHourSelector Is Nothing Then Exit Sub
        If objHourSelector.ErrorString = "" Then
                objHourSelector.ErrorString = "INVALID DATA TIME"
        Else
                objHourSelector.ErrorString = ""
        End If
End Sub
```

ExtSettingsFile, HourSelectorCmdTarget Property

Syntax ExtSettingsFile = _String

Description This property sets or returns the external configuration file for the referenced object. This file can also be specified in design mode in the object's 'Ext. File Settings' property. The extension provided for this file is ".SXML".

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
Dim objSymbol As HourSelectorCmdTarget
Set objSymbol = GetSynopticObject.GetSubObject("TestObject").GetObjectInterface
If objSymbol Is Nothing Then Exit Sub
```

```
objSymbol.ExtSettingsFile = "test.sxml"
objSymbol.SaveExtSettings
Set objSymbol= Nothing
End Sub
```

GraphicButtons, HourSelectorCmdTarget Property

Syntax GraphicButtons = _Boolean

Description When Enabling this property, the Hour Selector Window buttons are drawn using an icon instead of text. The text will instead be displayed as a tooltip when positioning the mouse on top of the button.

The tooltip is not managed in Windows CE versions.

This property is not managed by the 'Alarm Banner' object.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Click()
        GraphicButtons = True
        RecalcLayout
End Sub
```

GridModeBtnText, HourSelectorCmdTarget Property

Syntax GridModeBtnText = _String

Description This property sets or returns the text displayed in the "Grid Mode" button. The predefined text will display when setting this property with an empty string. Modifications only come into effect after the "RecalcLayout" has been called.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim objHourSelector As HourSelectorCmdTarget
```

```
        If objHourSelector Is Nothing Then Set objHourSelector =
        GetSynopticObject.GetSubObject("HourSelector1").GetObjectInterface
        If objHourSelector Is Nothing Then Exit Sub
        If objHourSelector.GridModeBtnText = "" Then
                objHourSelector.GridModeBtnText = "GRID MODE (F6)"
        Else
                objHourSelector.GridModeBtnText = ""
        End If
        objHourSelector.RecalcLayout
End Sub
```

HolidaysBtnText, HourSelectorCmdTarget Property

Syntax HolidaysBtnText = _String

Description This property sets or returns a text for the Scheduler's Holiday button. If you do not specify anything Movicon will use the default text.

Parameter	Description
None	None

Result String

Example:

```
Dim objSelector As HourSelectorCmdTarget
Public Sub Click()
        Debug.Print objSelector.HolidaysBtnText
End Sub
Public Sub SymbolLoading()
        Set objSelector =
        GetSynopticObject.GetSubObject("HourSelector").GetObjectInterface
        objSelector.HolidaysBtnText= "Festivi"
End Sub
```

MaxRow, HourSelectorCmdTarget Property

Syntax MaxRow = _Integer

Description This property sets or returns the grid's number or rows, corresponding to the number of timeframes that can be programmed for each day of the week. Changes only take effect after having called the "RecalcLayout" method.

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        Dim objHourSelector As HourSelectorCmdTarget
        If objHourSelector Is Nothing Then Set objHourSelector =
        GetSynopticObject.GetSubObject("HourSelector1").GetObjectInterface
        If objHourSelector Is Nothing Then Exit Sub

        Dim sMenuItems(0 To 2) As String
        Dim nSelectItem As Integer

        sMenuItems(0) = "Reset"
        sMenuItems(1) = "Increase"
        sMenuItems(2) = "Decrease"
        nSelectItem = ShowPopupMenu(sMenuItems, , , )

        Select Case nSelectItem
                Case 0 ' Reset
                        objHourSelector.MaxRow = 0
                Case 1 ' Increase
                        objHourSelector.MaxRow = objHourSelector.MaxRow + 1
                Case 2 ' Decrease
                        objHourSelector.MaxRow = objHourSelector.MaxRow - 1
        End Select
        objHourSelector.RecalcLayout
End Sub
```

NetworkBackupServerName, HourSelectorCmdTarget Property

Syntax NetworkBackupServerName = _String

Description This property sets or returns the name of any Network Backup Server used for getting data to display in the Historical Log window when the primary server, the one set in the 'NetowrkServerName'property is in timeout.

Parameter	Description
None	None

Result String

Example:

```
Dim objHourSelector As HourSelectorCmdTarget
Public Sub Click()
        Debug.Print objHourSelector .NetworkBackupServerName
End Sub
Public Sub SymbolLoading()
        Set objHourSelector =
        GetSynopticObject.GetSubObject("HourSelector").GetObjectInterface
End Sub
```

NetworkServer name,HourSelectorCmdTarget Property

Syntax NetworkServerName = _String

Description This property returns the name of any Network Server where data is to be retrieved for displaying in the Hour Selector Window.

Parameter	Description
None	None

Result String

Example:

```
Dim objSelector As HourSelectorCmdTarget
Public Sub Click()
        Debug.Print objSelector.NetworkServerName
End Sub
Public Sub SymbolLoading()
        Set objSelector =
        GetSynopticObject.GetSubObject("HourSelector").GetObjectInterface
End Sub
```

PromptPad, HourSelectorCmdTarget Property

Syntax PromptPad = _Boolean

Description When setting this property to "true" when the 'HourSelector is in "Grid" mode, the numeric or alphanumeric pad will be made available for editing the grid's values. Setting this property to "False" the values will become editable directly within the grid's cells.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
   Dim objHourSel As HourSelectorCmdTarget
   Set objHourSel = GetSynopticObject.GetSubObject("HourSel").GetObjectInterface
   If objHourSel Is Nothing Then Exit Sub
   If objHourSel .PromptPad = False Then
      objHourSel .PromptPad = True
   End If
End Sub
```

RemoveBtnText, HourSelectorCmdTarget Property

Syntax RemoveBtnText = _String

Description This property sets or returns the text displayed in the button used for removing schedulers added in runtime. The predefined text will display when this property is set with an empty string. Changes take effect only after the "RecalcLayout" method has been called.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim objHourSelector As HourSelectorCmdTarget
        If objHourSelector Is Nothing Then Set objHourSelector =
        GetSynopticObject.GetSubObject("HourSelector1").GetObjectInterface
        If objHourSelector Is Nothing Then Exit Sub
        If objHourSelector.RemoveBtnText = "" Then
                objHourSelector.RemoveBtnText = "REMOVE (CANC)"
        Else
                objHourSelector.RemoveBtnText = ""
        End If
        objHourSelector.RecalcLayout
End Sub
```

SaveBtnText, HourSelectorCmdTarget Property

Syntax SaveBtnText = _String

Description This property sets or returns a text for the Hour Selector's Save command button. If no text has been specialized, Movicon will use the default text.

Parameter	Description
None	None

Result String

Example:

```
Dim objSelector As HourSelectorCmdTarget
Public Sub Click()
        Debug.Print objSelector.SaveBtnText
End Sub
Public Sub SymbolLoading()
        Set objSelector =
        GetSynopticObject.GetSubObject("HourSelector").GetObjectInterface
```

```
        objSelector.SaveBtnText= "Salva"
End Sub
```

Scheduler, HourSelectorCmdTarget Property

Syntax Scheduler = _String

Description This property sets or returns the name of the Scheduler associated to the object.

Parameter	Description
None	None

Result String

Example:

```
Dim objSelector As HourSelectorCmdTarget
Public Sub Click()
        Debug.Print objSelector.Scheduler
End Sub
Public Sub SymbolLoading()
        Set objSelector =
        GetSynopticObject.GetSubObject("HourSelector").GetObjectInterface
End Sub
```

ShowAddBtn, HourSelectorCmdTarget Property

Syntax ShowAddBtn = _Boolean

Description This property sets or returns the visibility status of the button used for adding schedulers in runtime. Modifications take effect only after having called the "RecalcLayout" method.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim objHourSelector As HourSelectorCmdTarget
        If objHourSelector Is Nothing Then Set objHourSelector =
        GetSynopticObject.GetSubObject("HourSelector1").GetObjectInterface
        If objHourSelector Is Nothing Then Exit Sub
        objHourSelector.ShowAddBtn = Not objHourSelector.ShowAddBtn
        objHourSelector.RecalcLayout
End Sub
```

ShowCancelBtn, HourSelectorCmdTarget Property

Syntax ShowCancelBtn = _Boolean

Description This property shows the cancel button to cancel modifications made to the hour selector data.

Parameter	Description
None	None

Result Boolean

Example:

```
Dim objSelector As HourSelectorCmdTarget
Public Sub Click()
	If Not objSelector Is Nothing Then
		MsgBox "objSelector 's ShowCancelBtn is " & objSelector.ShowCancelBtn,vbInformation,GetProjectTitle
		objSelector.ShowCancelBtn= Not objSelector.ShowCancelBtn
		objSelector.RecalcLayout
	Else
		MsgBox "objSelector is nothing",vbInformation,GetProjectTitle
	End If
End Sub
Public Sub SymbolLoading()
	Set objSelector = GetSynopticObject.GetSubObject("HourSelector").GetObjectInterface
End Sub
```

ShowColumValue, HourSelectorCmdTarget Property

Syntax ShowColumValue = _Boolean

Description This property sets or returns the column's visibility status of the Grid used for setting the set valued in the command used by the scheduler. Modification go into effect only after the "RecalcLayout" has been called.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
	Dim objHourSelector As HourSelectorCmdTarget
	If objHourSelector Is Nothing Then Set objHourSelector = GetSynopticObject.GetSubObject("HourSelector1").GetObjectInterface
	If objHourSelector Is Nothing Then Exit Sub
```

```
        objHourSelector.ShowColumValue = Not objHourSelector.ShowColumValue
        objHourSelector.RecalcLayout
End Sub
```

ShowColumVariable, HourSelectorCmdTarget Property

Syntax ShowColumVariable = _Boolean

Description This property sets or returns the column's visibility status of the grid used for setting the variable name in the command used by the scheduler. Modifications go into effect only after the "RecalcLayout" method has been called.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim objHourSelector As HourSelectorCmdTarget
        If objHourSelector Is Nothing Then Set objHourSelector =
        GetSynopticObject.GetSubObject("HourSelector1").GetObjectInterface
        If objHourSelector Is Nothing Then Exit Sub
        objHourSelector.ShowColumVariable = Not
        objHourSelector.ShowColumVariable
        objHourSelector.RecalcLayout
End Sub
```

ShowComboScheduler, HourSelectorCmdTarget Property

Syntax ShowComboScheduler = _Boolean

Description This proeprty sets or returns the visibility status of the combo box used for selecting schedulers in runtime. Modifications go into effect only after the "RecalcLayout" method has been called.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim objHourSelector As HourSelectorCmdTarget
```

```
        If objHourSelector Is Nothing Then Set objHourSelector =
        GetSynopticObject.GetSubObject("HourSelector1").GetObjectInterface
        If objHourSelector Is Nothing Then Exit Sub
        objHourSelector.ShowComboScheduler = Not
        objHourSelector.ShowComboScheduler
        objHourSelector.RecalcLayout
End Sub
```

ShowHolidaysBtn, HourSelectorCmdTarget Property

Syntax ShowHolidaysBtn = _Boolean

Description This property allows a command button to be displayed for switching from the Scheduler's normal daily plan display to the holiday one and viceversa.

Parameter	Description
None	None

Result Boolean

Example:

```
Dim objSelector As HourSelectorCmdTarget
Public Sub Click()
        If Not objSelector Is Nothing Then
                MsgBox "objSelector 's ShowHolidaysBtn is " &
                objSelector.ShowHolidaysBtn,vbInformation,GetProjectTitle
                objSelector.ShowHolidaysBtn= Not objSelector.ShowHolidaysBtn
                objSelector.RecalcLayout
        Else
                MsgBox "objSelector is nothing",vbInformation,GetProjectTitle
        End If
End Sub
Public Sub SymbolLoading()
        Set objSelector =
        GetSynopticObject.GetSubObject("HourSelector").GetObjectInterface
End Sub
```

ShowRemoveBtn, HourSelectorCmdTarget Property

Syntax ShowRemoveBtn = _Boolean

Description This property sets or returns the visibility status of the button used for removing schedulers added in runtime. In cases where this property has been set with a new value, you will need to used the "Refresh" method.

Parameter	Description

None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim objHourSelector As HourSelectorCmdTarget
        If objHourSelector Is Nothing Then Set objHourSelector =
        GetSynopticObject.GetSubObject("HourSelector1").GetObjectInterface
        If objHourSelector Is Nothing Then Exit Sub
        objHourSelector.ShowRemoveBtn = Not objHourSelector.ShowRemoveBtn
        objHourSelector.RecalcLayout
End Sub
```

ShowSaveBtn, HourSelectorCmdTarget Property

Syntax ShowSaveBtn = _Boolean

Description This property shows the Save button for saving any changes made to the Hour Selector data.

Parameter	Description
None	None

Result Boolean

Example:

```
Dim objSelector As HourSelectorCmdTarget
Public Sub Click()
        If Not objSelector Is Nothing Then
                MsgBox "objSelector 's ShowSaveBtn is " &
                objSelector.ShowSaveBtn,vbInformation,GetProjectTitle
                objSelector.ShowSaveBtn= Not objSelector.ShowSaveBtn
                objSelector.RecalcLayout
        Else
                MsgBox "objSelector is nothing",vbInformation,GetProjectTitle
        End If
End Sub
Public Sub SymbolLoading()
        Set objSelector =
        GetSynopticObject.GetSubObject("HourSelector").GetObjectInterface
End Sub
```

StartTimeColText, HourSelectorCmdTarget Property

Syntax StartTimeColText = _String

Description This property sets or returns the text displayed in the title of the first grid column. When setting this property with an empty string, the predefined text will be displayed. Modifications will go into effect only after the "RecalcLayout" method has been called.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim objHourSelector As HourSelectorCmdTarget
        If objHourSelector Is Nothing Then Set objHourSelector =
        GetSynopticObject.GetSubObject("HourSelector1").GetObjectInterface
        If objHourSelector Is Nothing Then Exit Sub
        If objHourSelector.StartTimeColText = "" Then
                objHourSelector.StartTimeColText = "START"
        Else
                objHourSelector.StartTimeColText = ""
        End If
        objHourSelector.RecalcLayout
End Sub
```

ValueColText, HourSelectorCmdTarget Property

Syntax ValueColText = _String

Description This property sets or returns the text displayed in the grid's forth column. The predefined text will be displayed when setting this property with an empty string. Modification take effect only after the "RecalcLayout" method has been called.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim objHourSelector As HourSelectorCmdTarget
        If objHourSelector Is Nothing Then Set objHourSelector =
        GetSynopticObject.GetSubObject("HourSelector1").GetObjectInterface
        If objHourSelector Is Nothing Then Exit Sub
```

```
        If objHourSelector.ValueColText = "" Then
                objHourSelector.ValueColText = "VALUE"
        Else
                objHourSelector.ValueColText = ""
        End If
        objHourSelector.RecalcLayout
End Sub
```

ValueErrorString, HourSelectorCmdTarget Property

Syntax ValueErrorString= _String

Description This property sets or resets the error message set in the HourSelector object when the min. and max. limits, set in a variable whose value is modified through the HourSelector's cell value, are exceeded.

Parameter	Description
None	None

Result String

Example:

```
'per leggere il contenuto
Public Sub Click()
Dim objSelector As HourSelectorCmdTarget
Set objSelector = GetSynopticObject.GetSubObject("hs").GetObjectInterface
If Not objSelector Is Nothing Then
    MsgBox "objSelector 's ValueErrorString is " & objSelector.valueerrorstring
Else
    MsgBox "objSelector is nothing",vbInformation,GetProjectTitle
End If
End Sub

'per scrivere il contenuto
Public Sub Click()
Dim objSelector As HourSelectorCmdTarget
Set objSelector = GetSynopticObject.GetSubObject("hs").GetObjectInterface
If Not objSelector Is Nothing Then
  objSelector.valueerrorstring = "nuovo valore impostato"
End If
End Sub
```

VariableColText, HourSelectorCmdTarget Property

Syntax VariableColText = _String

Description This property sets or returns the text displayed in the gird's third column's title. Setting this property with an empty string will display the predefined text. Changes only go into effect after the "RecalcLayout" method has been called.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim objHourSelector As HourSelectorCmdTarget
        If objHourSelector Is Nothing Then Set objHourSelector =
        GetSynopticObject.GetSubObject("HourSelector1").GetObjectInterface
        If objHourSelector Is Nothing Then Exit Sub
        If objHourSelector.VariableColText = "" Then
                objHourSelector.VariableColText = "TAG"
        Else
                objHourSelector.VariableColText = ""
        End If
        objHourSelector.RecalcLayout
End Sub
```

36. IOPortInterface

IOClosePort, IOPortInterface Function

Syntax IOClosePort(_lPortID)

Description Closes the communication port identified by the "lPortID" parameter.

The function's returned value may be:
-0: communication port closed correctly
-1: "lPortID" unknown

Parameter	Description
lPortID As Long	Open serial port identification

Result Long

Example:

```
Sub Main
        Dim objIOPort As IOPortInterface
        Dim bResult As Boolean
        Set objIOPort = GetIOPortInterface
        Debug.Print objIOPort.IOClosePort(PortID)
        Set objIOPort = Nothing
End Sub
```

IOGetLastError, IOPortInterface Function

Syntax IOGetLastError(_lPortID)

Description Gets the last communication error. This property is not available in the programming mode and only in read in Runtime mode. This method does not get the last function return error code (eg. -6). It gets any additional error which might help to analyze the problem: for example if no value is returned when using the IOInput you could call the IOGetLastError to find out if and what error has been made.

The functions returned values have the following meanings:

0x0001 = Input buffer overflow. The input buffer has run out of space.
0x0002 = Port overrun. The hardware did not read a character before the arrival of the next one and therefore the character was lost.
0x0004 = Parity error in input. A parity error has been returned.
0x0008 = Input frame error. An error in the input frame has been returned.
0x0100 = Output Buffer full. The Output buffer space has run out while a character was being entered.
0x0200 = Printer TimOut error. A Timeout error has been found in the Printer.
0x0400 = I/O Printer Error. An error made by the printer has been found.

0x0800 = Printer error. Device has not been selected.
0x1000 = Printer error. The printer has run out of paper.
0x8000 = Error made when request for an unsupported mode was made.

Parameter	Description
lPortID As Long	Open Serial Port identification.

Result Long

Example:

```
Sub Main
        Dim objIOPort As IOPortInterface
        Dim sVariant As String
        Dim PortID  As Long
        Set objIOPort = GetIOPortInterface
        PortID = objIOPort.IOPortOpen("COM1:9600,n,8,1")
        MsgBox "IOGetLastError = " & objIOPort.IOGetLastError(PortID), vbInformation,
        GetProjectTitle
        Set objIOPort = Nothing
End Sub
```

IOInBufferCount, IOPortInterface Function

Syntax IOInBufferCount(_lPortID)

Description Gets the number of characters waiting in the Input Buffer. Not available in programming mode.
The IOInBufferCount indicates the number of characters received by the modem and temporarily stored in the input buffer. The buffer can be flushed by setting the IOInBufferCount property to '0'.

Parameter	Description
lPortID As Long	Open serial port identification

Result Integer

Example:

```
Sub Main
        Dim objIOPort As IOPortInterface
        Dim ID As Long
        Set objIOPort = GetIOPortInterface
        ID = objIOPort.IOPortOpen("COM1:9600,n,8,1")
        MsgBox "IOInBufferCount = " & objIOPort.IOInBufferCount(ID), vbInformation,
        GetProjectTitle
        objIOPort.IOInBufferCount(ID) = 0
        Set objIOPort = Nothing
End Sub
```

IOInput, IOPortInterface Function

Syntax IOInput(_lPortID, _bMode)

Description Gets and moves data flow from the input buffer. Not available in programming mode and available in Runtime mode in read only. The bMode parameter determines how the data is to be retrieved:

False = Input function returns text data in a Variant variable
True = Input property returns binary data in an array of bytes in a Variant variable.

Tip:a) IOInputLen = 0: IOInput reads the serial's reception buffer contents for a maximum number of 255 characters. Then if the buffer contains more than 255 characters, you will need to repeat this function a few times as necessary. 'IOInBufferCount' updates each time characters are read by the serial's buffer with the IOInput.
b) IOInputLen <> 0 (max. 255): IOInput gets the number of characters set with 'IOInputLen' property only when the serial's buffer contains an equal or major number of characters requests; otherwise returns with an empty variant (VT_EMPTY).

Parameter	Description
lPortID As Long bMode As Boolean	Open serial port identification. data retrieval mode.

Result Long

Example:

```
Sub Main
        Dim objIOPort As IOPortInterface
        Dim i As Integer
        Dim bByte As Variant
        Dim sDebug As String
        Set objIOPort = GetIOPortInterface
        bByte = objIOPort.IOInput(PortID, True)
        For i = 0 To UBound(bByte)
        sDebug = sDebug & bByte(i) & ","
        Next i
        Set objIOPort = Nothing
        Debug.Print sDebug
End Sub
```

IOOutput, IOPortInterface Function

Syntax IOOutput(_lPortID, _vData)

Description Writes data in the output buffer. This function is not available in programming mode and only in read in Runtime mode.

The IOOutput property allows text or binary data to be transmitted. To send text data with the IOOutput property you need to specify a variant variable which includes a string. (The variable should therefore be managed as a string type variable). To enter binary data you will need to pass a Variant variable to the IOOutput property containing a byte matrix (the variable whoudl therefore be managed as a byte array variable).
The ANSI strings are generally entered as text data. Data which includes control characters, such as the NULL, characters, are sent as binary data,

therefore can be sent only as byte arrays.

Parameter	Description
vData As Variant	Data to be sent.
lPortID As Long	identifies the open serial port

Result Long
0: Funcion executed successfully.
-1: Mismatch type in Parameter. Pass a one-dimensional array
-2: Failed getting the lowest index of the array
-3: Failed getting the highest index of the array
-4: Failed getting array pointer
-5: Generic Error.

Example:

```
Sub Main
        Dim objIOPort As IOPortInterface
        Dim sVariant As String
        Set objIOPort = GetIOPortInterface
        sVariant = "ABC"
        Debug.Print objIOPort.IOOutput(PortID, sVariant)
        Set objIOPort = Nothing
End Sub
```

IOPortOpen, IOPortInterface Function

Syntax IOPortOpen(_lpszPortSettings)

Description Opens the communication port by using the settings specified in the parameter contained in "lpszPortSettings".

The value returned by the function identifies the open port. This Long type value is needed in order to use the other methods and properties of the IOPortInterface. This function may also return values indicating errors as follows:

-1: The number of characters in the lpszPortSettings parameter is less than 4
-2: Failed to open serial port
-3: Port settings could not be interpreted (eg. wrong baud-rate, etc)
-4: Errors made while setting the values in the serial port (eg. baud-rate, etc.)

Example: COM1:9600,n,8,1
If you enter only "COM1" as the serial port's settings, the Windows default settings will be used instead.

Parameter	Description
lpszPortSettings As String	Settings for the Serial port.

Result Long

Example:

```
Sub Main
        Dim objIOPort As IOPortInterface
        Dim ID As Long
        Set objIOPort = GetIOPortInterface
        ID = objIOPort.IOPortOpen("COM1:9600,n,8,1")
        Set objIOPort = Nothing
        MsgBox "PortOpen = " & ID, vbInformation, GetProjectTitle
End Sub
```

IOBreak, IOPortInterface Property

Syntax IOBreak(_lPortID)

Description This highers or lowers the BREAK signal in the open serial port. This property is not available in programming mode.
The True value enables the Break status, the False value disables the Break status. For further information on BREAK signals please consult the UART guide starting from 8250 and onwards.

Parameter	Description
lPortID As Long	ID of the open serial port.

Result Boolean

Example:

```
Sub Main
        Dim objIOPort As IOPortInterface
        Set objIOPort = GetIOPortInterface
        objIOPort.IOBreak(PortID) = True
        Set objIOPort = Nothing
End Sub
```

IOCDHolding, IOPortInterface Property

Syntax IOCDHolding(_lPortID)

Description Determines whether a carrier is present, by checking the CD line status (Carrier Detect). The CD signal is sent from a modem to its computer to indicate that the modem is ready for transmitting. This property is not available in programming mode but is available in Runtime mode in read only.
The returned values are:

True = the CD line is active
False = the CD line is not active

Tip: It is important to monitor Carrier loss especially when using host applications, such as BBS, where the transmission may be interrupted by the dialer (holding loss) at anytime. The CD condition is also called RLSD (Receive Line Signal Detect).

Parameter	Description
lPortID As Long	Identification of open serial port.

Result Boolean

Example:

```
Sub Main
        Dim objIOPort As IOPortInterface
        Set objIOPort = GetIOPortInterface
        objIOPort.IOCDHolding(PortID) = True
        Set objIOPort = Nothing
End Sub
```

IIOCTSHolding, IOPortInterface Property

Syntax IOCTSHolding(_lPortID)

Description Determines whether data can be sent by verifying the CTS line status (Clear To Send). The CTS signal is usually sent from a modem to its computer to indicate that transmission can proceed. This property is not available in programming mode and available in Runtime mode in read only.
The returned value may be:

True = the CTS line is active
False = the CTS line is not active

Tip: The CTS line is used for synchonous RTS/CTS (Request To Send/Clear To Send) hardware. The IOCTSHolding property allows the CTS line polling to be executed manually to identify the status.

Parameter	Description
lPortID As Long	Open serial port identification

Result Boolean

Example:

```
Sub Main
        Dim objIOPort As IOPortInterface
        Dim bResult As Boolean
        Set objIOPort = GetIOPortInterface
        bResult = objIOPort.IOCTSHolding(PortID)
        MsgBox "CTSHolding = " & bResult
        Set objIOPort = Nothing
End Sub
```

IODSRHolding, IOPortInterface Property

Syntax IODSRHolding(_lPortID)

Description Determines the status of a DSR line (Data Set Ready). A modem usually sends the DSR signal to the computer to which it is attached, to indicate that it is ready to operate. This property is not available in programming mode and available only in read in Runtime mode.
The returned values are:

True = DSR line is active
False = DSR line is not active

Tip: This property is useful for writing Data Set Ready/Data Terminal Ready synchronous routines for DTE (Data Terminal Equipment) computers.

Parameter	Description
lPortID As Long	Open serial port identification

Result Boolean

Example:

```
Sub Main
        Dim objIOPort As IOPortInterface
        Dim bResult As Boolean
        Set objIOPort = GetIOPortInterface
        bResult = objIOPort.IODSRHolding(PortID)
        MsgBox "DSRHolding= " & bResult
        Set objIOPort = Nothing
End Sub
```

IODTREnable, IOPortInterface Property

Syntax IODTREnable(_lPortID)

Description Determines whether to activate the DTR line (Data Terminal Ready) during communications. The DTR signal is sent from a computer to its modem to indicate that the computer is ready to accept incoming transmissions.
The True value enables the DTR line while the False value disables it.

Tip: When the IODTREnable is set to True, the DTR line is activated (on) when the port is opened and deactivates (off) when the port is closed. When the IODTREnable is set to False, the DTR line is always deactivated. In most cases, the deactivation of the DTR line is equivalent to hanging up the telephone.

Parameter	Description
lPortID As Long	Open serial port identification

Result Boolean

Example:

```
Sub Main
        Dim objIOPort As IOPortInterface
        Dim bResult As Boolean
        Set objIOPort = GetIOPortInterface
        objIOPort.IODTREnable(PortID) = True
        Set objIOPort = Nothing
End Sub
```

IOInputLen, IOPortInterface Property

Syntax IOInputLen(_lPortID)

Description Sets or returns the number of characters that the Input function can read from the input buffer.

Tip: a) IOInputLen = 0: IOInput reads the serial's reception buffer contents for a maximum number of 255 characters. Then if the buffer contains more than 255 characters, you will need to repeat this function a few times as necessary. 'IOInBufferCount' updates each time characters are read by the serial's buffer with the IOInput.
b) IOInputLen <> 0 (max. 255): IOInput gets the number of characters set with 'IOInputLen' property only when the serial's buffer contains an equal or major number of characters requests; otherwise returns with an empty variant (VT_EMPTY).

Parameter	Description
lPortID As Long	Open serial port identity

Result Integer

Example:

```
Sub Main
        Dim objIOPort As IOPortInterface
        Dim nValue As Integer
        Dim PortID As Long
        Set objIOPort = GetIOPortInterface
        PortID = objIOPort.IOPortOpen("COM1:9600,n,8,1")
        nValue = Val(InputBox("0 = Max.", "InputLen", "0", 200, 200))
        objIOPort.IOInputLen(PortID) = nValue
        Set objIOPort = Nothing
End Sub
```

IOOutBufferCount, IOPortInterface Property

Syntax IOOutBufferCount(_lPortID)

Description Returns the number of characters waiting in the output buffer. It can also be used for flushing the output buffer. This property is not available in programming mode.
To flush the input buffer, set the IOOutBufferCount to '0'.

Parameter	Description
lPortID As Long	identifies open serial port

Result Integer

Example:

```
Sub Main
        Dim objIOPort As IOPortInterface
        Set objIOPort = GetIOPortInterface
        MsgBox "IOOutBufferCount = " & objIOPort.IOOutBufferCount(PortID), vbInformation,
        GetProjectTitle
        Set objIOPort = Nothing
End Sub
```

IORTSEnable, IOPortInterface Property

Syntax IORTSEnable(_lPortID)

Description Enables or disables the RTS line (Request to Send). The RTS signal is usually sent from the computer to its modem. The True value activates the RTS line and the False value deactivates it.

Tip: When the IORTSEnable is set to True, the RTS line is activated (on) when the port is opened, and deactivated (off) when the port is closed.
The RTS line is used for synchronous RTS/CTS hardware. The IORTSEnable property allows manual polling of the RTS line to identify the line's status.

Parameter	Description
lPortID As Long	Identifies the open serial port

Result Boolean

Example:

```
Sub Main
        Dim objIOPort As IOPortInterface
        Set objIOPort = GetIOPortInterface
        objIOPort.IORTSEnable(PortID) = True
        Set objIOPort = Nothing
End Sub
```

37. ListBoxCmdTarget

OnSelected, ListBoxCmdTarget Event

Description Event occurs each time an item from the list is selected.

Parameter	Description
nSel As Long	Selected item's index

OnSelecting, ListBoxCmdTarget Event

Description Event occurs each an item from the list is selected.

Parameter	Description
nSel As Long	Selected item's index
bRet As boolean	Enable upon selection

AddString, ListBoxCmdTarget Function

Syntax AddString(_lpszItem)

Description This function adds the string passed with the lpszItem parameter to the list. The function returns the string's position on the list.

Parameter	Description
lpszItem As String	String to be added to the list

Result Long

Example:

```
Public Sub Click()
        Debug.Print GetObjectInterface.AddString("stringa1")
End Sub
```

GetCount, ListBoxCmdTarget Function

Syntax GetCount()

Description This function returns the number of strings inserted in the list.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Debug.Print GetObjectInterface.GetCount
End Sub
```

GetSelectedIndex, ListBoxCmdTarget Function

Syntax GetSelectedIndex()

Description This function returns the index of the string selected from the list.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Debug.Print GetObjectInterface.GetSelectedIndex
End Sub
```

GetText, ListBoxCmdTarget Function

Syntax GetText(_nIndex)

Description This function gets the text relating to the position on the list specified by the nIndex parameter.

Parameter	Description

nIndex As Long	List index of the text to be retrieved.

Result String

Example:

```
Public Sub Click()
        Debug.Print GetObjectInterface.GetText(GetObjectInterface.GetCount-1)
End Sub
```

LoadExtSettings, ListBoxCmdTarget Function

Syntax LoadExtSettings

Description This function allows you to load the object's setting from the relative external setting file. This file can be specified in the "Settings File" property during design mode or by using the "ExtSettingsFile" interface property. This extension provided for this file is ".SXML".

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
Dim objSymbol As ListBoxCmdTarget
Set objSymbol = GetSynopticObject.GetSubObject("TestObject").GetObjectInterface
If objSymbol Is Nothing Then Exit Sub
objSymbol.ExtSettingsFile = "test.sxml"
objSymbol.LoadExtSettings
Set objSymbol = Nothing
End Sub
```

RefillList, ListBoxCmdTarget Function

Syntax RefillList()

Description This function updates the list contents.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Debug.Print GetObjectInterface.RefillList
End Sub
```

RemoveString, ListBoxCmdTarget Function

Syntax RemoveString(_lpszItem)

Description This function removes the string passed with the pszItem parameter from the list. This function returns the true boolean value when the operation is successful.

Parameter	Description
lpszItem As String	String to be removed from the list

Result Boolean

Example:

```
Public Sub Click()
        Debug.Print GetObjectInterface.RemoveString("stringa1")
End Sub
```

SaveExtSettings, ListBoxCmdTarget Function

Syntax SaveExtSettings

Description This function allows you to save the the object's configuration in the relating external settings file. This file can be specified in design mode in the "Ext. File Settings property", or using the "ExtSettingsFile" interface property. The extension to use for this file is ".SXML".

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
Dim objSymbol As ListBoxCmdTarget
Set objSymbol = GetSynopticObject.GetSubObject("TestObject").GetObjectInterface
If objSymbol Is Nothing Then Exit Sub
objSymbol.ExtSettingsFile = "test.sxml"
objSymbol.SaveExtSettings
Set objSymbol = Nothing
End Sub
```

ExtSettingsFile, ListBoxCmdTarget Property

Syntax ExtSettingsFile = _String

Description This property sets or returns the external configuration file for the referenced object. This file can also be specified in design mode in the object's 'Ext. File Settings' property. The extension provided for this file is ".SXML".

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim objSymbol As ListBoxCmdTarget
        Set objSymbol = GetSynopticObject.GetSubObject("TestObject").GetObjectInterface
        If objSymbol Is Nothing Then Exit Sub
        objSymbol.ExtSettingsFile = "test.sxml"
        objSymbol.SaveExtSettings
        Set objSymbol= Nothing
End Sub
```

ListData, ListBoxCmdTarget Property

Syntax ListData = _String

Description This property sets or returns the ListBox contents, being the string set in the list or combo box object's "ListBox Item" property. When the "ListData" property's value is changed you will need to use the ""RefillList" method to update the list in the object.

Caution: When the list is dynamic, therefore retrieved from the string variable linked to the object's "Var. List ListBox" field, the "ListData" property is in read only and always returns the string variable's contents.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Debug.Print GetObjectInterface.ListData
End Sub
```

ListVariable, ListBoxCmdTarget Property

Syntax ListVariable = _String

Description This property sets or gets the name of the variable linked to the list of values inserted in the Movicon ListBox.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Debug.Print GetObjectInterface.ListVariable
End Sub
```

SortItems,ListBoxCmdTarget Property

Syntax SortItems = _String

Description Consents you to set or read the property that is used for sorting the elements in the object.

Parameter	Description
None	None

Result String

Example1:

```
'Using a Combo-Box object:

Dim objCombo As DrawCmdTarget
Dim objDisplay As DisplayEditCmdTarget
Dim objList As ListBoxCmdTarget
Public Sub Click()
        Set objCombo=GetSynopticObject.GetSubObject("combobox")
        Set objDisplay=objCombo.GetObjectInterface()
        Set objList=objDisplay.GetComboListInterface
        objList.SortItems=True
        objList.RefillList
        Set objCombo=Nothing
        Set objDisplay=Nothing
        Set objList=Nothing
End Sub
```

Example2:

```
'Using a List-Box object (write):

Dim objListBox As ListBoxCmdTarget
Public Sub Click()
```

```
        Set objListBox=GetSynopticObject.GetSubObject("listbox").GetObjectInterface
        objListBox.SortItems=True
        objListBox.RefillList
        Set objListBox=Nothing
End Sub
```

Example3:

```
'Using a List-Box object (read):

Dim objListBox As ListBoxCmdTarget
Public Sub Click()
        Set objListBox=GetSynopticObject.GetSubObject("listbox").GetObjectInterface
        Dim sorted As Boolean
        sorted=objListBox.SortItems
        debug.print "SortItems value = " & sorted
        Set objListBox=Nothing
End Sub
```

Variable, ListBoxCmdTarget Property

Syntax Variable = _String

Description This property sets or gets the name of the variable linked to the item selected from Movicon ListBox.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Debug.Print GetObjectInterface.Variable
End Sub
```

38. NetworkClientCmd

ConnectVariable, NetworkClientCmd Function

Syntax ConnectVariable(_lpszVarName, _lpszServerName, _lpszServerVarName, _nMode)

Description This function allows a Client side variable to be connected to a variable on the server.

The connection modes are:

0 input
1 output
2 input/output

Parameter	Description
lpszVarName As String	Name of the variable to be connected
lpszServerName As String	Name of the Server to be connected to
lpszServerVarName As String	Name of the variable on the Server side
nMode As Integer	Connection mode
bSynchronous as boolean	allows you to choose whether to connect the variable in synchronous or asynchronous mode (default = True). bSynchronous = True : this function returns when the variable has connected to the server and its value read. Therefore this function waits until the timeout set in the client has run out before connecting each variable in input/output or input. bSynchronous = False : this function connects the variable to the server and returns straight away without waiting to see that the variable contains a valid value.

Result Boolean

Example:

```
Public Sub Click()
        Dim NetwObj As NetworkClientCmd
        Set NetwObj  = GetNetworkClient
        If Not NetwObj Is Nothing Then
                NetwObj.ConnectVariable("VAR00001", "Server2", "VAR00005", 0,true)
                Set NetwObj = Nothing
        End If
End Sub
```

DisconnectVariable, NetworkClientCmd Function

Syntax DisconnectVariable(_lpszVarName)

Description This function allows a variable to be disconnected on the client side by the server.

Parameter	Description
lpszVarName As String	Name of variable to be disconnected

Result Boolean

Example:

```
Public Sub Click()
        Dim NetwObj As NetworkClientCmd
        Set NetwObj  = GetNetworkClient
        If Not NetwObj Is Nothing Then
                NetwObj.DisconnectVariable("VAR00001")
                Set NetwObj = Nothing
        End If
End Sub
```

GetClientRules, NetworkClientCmd Function

Syntax GetClientRules(_lpszClientRulesName)

Description This functiopn allows you to get the ClientRules object, which is part of the ClientRuleInterface, referenced by the nam passed as parameter. When an empty string is passed as the parameter, the object will refer to the general properties of the Network Client. When the parameter contains a name of a valid Client Rules, the object will refer to its general properties.

Parameter	Description
lpszClientRulesName As String	Name of ClientRules object to be retrieved

Result Object
If Function has been executed successfully it will retrieve an object of type ClientRulesInterface if otherwise Nothing is returned.

Example:

```
Public Sub Click()
        Dim NetwObj As NetworkClientCmd
        Dim ClientObj As ClientRulesInterface
        Set NetwObj  = GetNetworkClient
        If Not NetwObj Is Nothing Then
                Set ClientObj = NetwObj.GetClientRules("Server2")
                If Not ClientObj Is Nothing Then
                        Debug.Print ClientObj.ClientTimeout
                        Set ClientObj = Nothing
                End If
                Set NetwObj = Nothing
```

```
        End If
End Sub
```

GetRASStation, NetworkClientCmd Function

Syntax GetRASStation(_lpszRASStation)

Description This function allows you to get the RASStation object, which is part of the RASStationInterface, referenced by the name passed as parameter.

Parameter	Description
lpszRASStation As String	RASStation object to be retrieved

Result Object
If Function has been executed successfully it will retrieve an object of type RASStationInterface if otherwise Nothing is returned.

Example:

```
Public Sub Click()
        Dim NetwObj As NetworkClientCmd
        Dim RASSObj As RASStationInterface
        Set NetwObj  = GetNetworkClient
        If Not NetwObj Is Nothing Then
                Set RASSObj = NetwObj.GetRASStation("RAS1")
                If Not RASSObj Is Nothing Then
                        Debug.Print RASStation.IsConnected
                        Set RASSObj = Nothing
                End If
                Set NetwObj = Nothing
        End If
End Sub
```

IsServerAvailable, NetworkClientCmd Function

Syntax IsServerAvailable(_lpszServerName)

Description This function verified whether the server, whose name is passed as parameter, is available for sharing variables.

Parameter	Description
lpszServerName As String	Name of the Server to be tested.

Result Boolean

Example:

```
Public Sub Click()
        Dim NetwObj As NetworkClientCmd
        Set NetwObj  = GetNetworkClient
        If Not NetwObj Is Nothing Then
                L$ = InputBox$("Enter server name:","","Server2")
```

```
            MsgBox "NetwObj.IsAvailable = " & NetwObj.IsServerAvailable(L),vbOkOnly,""
            Set NetwObj = Nothing
        End If
End Sub
```

39. NetworkRedudancyCmd

ActNumRetries, NetworkRedudancyCmd Function

Syntax ActNumRetries()

Description This function returns the actual number of connection retries made by the Secondary to the primary. The script is to be executed on the Secondary: when the primary crashes, the Secondary will carry out a number of retries equal to the valued inserted in the "# Retries" property, and this method returns the n° of connection retries being carried out.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim ObjRed As NetworkRedudancyCmd
        Set ObjRed = GetNetworkRedudancy
        If  ObjRed is Nothing Then Exit Sub
        MsgBox "ActNumRetries -> " & ObjRed.ActNumRetries
End Sub
```

CallBackServer, NetworkRedudancyCmd Function

Syntax CallBackServer()

Description This function returns further information on the server which the Secondary is connected to (Ip, port number, etc..).

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim ObjRed As NetworkRedudancyCmd
        Set ObjRed = GetNetworkRedudancy
```

```
        If  ObjRed is Nothing Then Exit Sub
        MsgBox "CallBackServer -> " & ObjRed.CallBackServer
End Sub
```

ConnectToServer, NetworkRedudancyCmd Function

Syntax ConnectToServer()

Description This function returns the Primary Server's IP address to which the Secondary is connected.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim ObjRed As NetworkRedudancyCmd
        Set ObjRed = GetNetworkRedudancy
        If  ObjRed is Nothing Then Exit Sub
        MsgBox "PrimaryServer IP -> " & ObjRed.ConnectToServer
End Sub
```

GetTotalPendingMessage, NetworkRedudancyCmd Function

Syntax GetTotalPendingMessage()

Description This function returns the actual number of messages waiting to be sent between the 2 Servers. The script can executed o both Servers where each one will return the value of its messages waiting to be sent.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim ObjRed As NetworkRedudancyCmd
        Set ObjRed = GetNetworkRedudancy
        If  ObjRed is Nothing Then Exit Sub
        MsgBox "GetTotalPendingMessage -> " & ObjRed.GetTotalPendingMessages
End Sub
```

IsActiveServer, NetworkRedudancyCmd Function

Syntax IsActiveServer()

Description This function returns the True of False boolean values according to the whether the Server, where the script is being run, is Active.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim ObjRed As NetworkRedudancyCmd
        Set ObjRed = GetNetworkRedudancy
        If  ObjRed is Nothing Then Exit Sub
        MsgBox "IsActiveServer -> " & ObjRed.IsActiveServer
End Sub
```

LastInteractionTime, NetworkRedudancyCmd Function

Syntax LastInteractionTime()

Description This function returns the date and time of when the last interaction took place between the two Servers; the script can be run on both Servers.

Parameter	Description
None	None

Result Date

Example:

```
Public Sub Click()
        Dim ObjRed As NetworkRedudancyCmd
        Set ObjRed = GetNetworkRedudancy
        If  ObjRed is Nothing Then Exit Sub
        MsgBox "LastInteractionTime -> " & ObjRed.LastInteractionTime
End Sub
```

PendingStartedDriverOnSecondary, NetworkRedudancyCmd Function

Syntax PendingStartedDriverOnSecondary()

Description This function returns the True boolean value when the Communication Drivers are waiting to be activated on the Secondary Server.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim ObjRed As NetworkRedudancyCmd
        Set ObjRed = GetNetworkRedudancy
        If  ObjRed is Nothing Then Exit Sub
        MsgBox "PendingStartedDriverOnSecondary -> " &
        ObjRed.PendingStartedDriverOnSecondary
End Sub
```

SecondaryServerConnected, NetworkRedudancyCmd Function

Syntax SecondaryServerConnected()

Description This function returns the True boolean valoue when the secondary server is connected to the primary.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim ObjRed As NetworkRedudancyCmd
        Set ObjRed = GetNetworkRedudancy
        If  ObjRed is Nothing Then Exit Sub
        MsgBox "SecondaryServerConnected -> " & ObjRed.SecondaryServerConnected
End Sub
```

StartedDriverOnSecondary, NetworkRedudancyCmd Function

Syntax StartedDriverOnSecondary()

Description This function returns the True boolean value when the communication drivers are active on the secondary.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim ObjRed As NetworkRedudancyCmd
        Set ObjRed = GetNetworkRedudancy
        If  ObjRed is Nothing Then Exit Sub
        MsgBox "StartedDriverOnSecondary -> " & ObjRed.StartedDriverOnSecondary
End Sub
```

StatusVariable, NetworkRedudancyCmd Function

Syntax StatusVariable()

Description This function returns the name of the variable set as the redundancy's status variable.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim ObjRed As NetworkRedudancyCmd
        Set ObjRed = GetNetworkRedudancy
        If  ObjRed is Nothing Then Exit Sub
        MsgBox "StatusVariable Name -> " & ObjRed.StatusVariable
End Sub
```

Synchronizing, NetworkRedudancyCmd Function

Syntax Synchronizing()

Description This function returns the True boolean value when the servers are in synchronizing mode. If the servers are not synchronized or have already been synchronized, the result will be False.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim ObjRed As NetworkRedudancyCmd
        Set ObjRed = GetNetworkRedudancy
        If  ObjRed is Nothing Then Exit Sub
        MsgBox "Synchronizing -> " & ObjRed.Synchronizing
End Sub
```

Type, NetworkRedudancyCmd Function

Syntax Type()

Description This function returns a number which indicates the Server type.

The possible values are:

0 none
1 primary
2 secondary

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        Dim ObjRed As NetworkRedudancyCmd
        Dim StrType As String
        Set ObjRed = GetNetworkRedudancy
        If  ObjRed is Nothing Then Exit Sub
        Select Case ObjRed.Type
                Case 0
                        StrType= "None"
                Case 1
                        StrType= "Primary"
                Case 2
                        StrType= "Secondary"
```

```
            Case Else
        End Select
        MsgBox "ServerType-> " & StrType
End Sub
```

DriverErrorTimeout, NetworkRedudancyCmd Property

Syntax DriverErrorTimeout = _Long

Description This property allows you to set a communication driver error timeout value (in ms).

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim ObjRed As NetworkRedudancyCmd
        Dim StrValue As String
        Set ObjRed = GetNetworkRedudancy
        If  ObjRed is Nothing Then Exit Sub
        StrValue = InputBox "Insert DriverErrorTimeout", "Input Value"
        ObjRed.DriverErrorTimeout = Val(StrValue)
        MsgBox "DriverErrorTimeout -> " & ObjRed.DriverErrorTimeout
End Sub
```

MaxHisCacheHits, NetworkRedudancyCmd Property

Syntax MaxHisCacheHits = _Long

Description This property sets or returns the maximum number of recordings before the records start being deleted.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim ObjRed As NetworkRedudancyCmd
        Dim StrValue As String
        Set ObjRed = GetNetworkRedudancy
        If  ObjRed is Nothing Then Exit Sub
        StrValue = InputBox "Insert MaxHisCacheHit", "Input Value"
```

```
        ObjRed.MaxHisCacheHits = Val(StrValue)
        MsgBox "MaxHisCacheHit -> " & ObjRed.MaxHisCacheHits
End Sub
```

Retries, NetworkRedudancyCmd Property

Syntax Retries = _Long

Description This property sets or returns the number of Secondary reconnection retries when the Primary crashes.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim ObjRed As NetworkRedudancyCmd
        Dim StrValue As String
        Set ObjRed = GetNetworkRedudancy
        If  ObjRed is Nothing Then Exit Sub
        StrValue = InputBox "Insert Retries", "Input Value"
        ObjRed.Retries= Val(StrValue)
        MsgBox "Retries -> " & ObjRed.Retries
End Sub
```

SwitchServerOnDriverError, NetworkRedudancyCmd Property

Syntax SwitchServerOnDriverError = _Boolean

Description This property permits the secondary (if set with the True boolean value) to activate the communication drivers if those on the primary fail to work.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim ObjRed As NetworkRedudancyCmd
        Dim StrValue As String
        Set ObjRed = GetNetworkRedudancy
        If  ObjRed is Nothing Then Exit Sub
        ObjRed.SwitchServerOnDriverError = Not ObjRed.SwitchServerOnDriverError
        MsgBox "SwitchServerOnDriverError -> " & ObjRed.SwitchServerOnDriverError
End Sub
```

SyncTimeFreq, NetworkRedudancyCmd Property

Syntax SyncTimeFreq = _Long

Description This property sets or returns the number of synchronizations needed between the two servers within 24 hours.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim ObjRed As NetworkRedudancyCmd
        Dim StrValue As String
        Set ObjRed = GetNetworkRedudancy
        If  ObjRed is Nothing Then Exit Sub
        StrValue = InputBox "Insert SyncTimeFreq ", "Input Value"
        ObjRed.SyncTimeFreq = Val(StrValue)
        MsgBox "Retries -> " & ObjRed.SyncTimeFreq
End Sub
```

TimeOut, NetworkRedudancyCmd Property

Syntax TimeOut = _Long

Description This property sets or returns the timeout (in ms) which the Secondary must wait before coming active when the primary crashes.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim ObjRed As NetworkRedudancyCmd
        Dim StrValue As String
        Set ObjRed = GetNetworkRedudancy
        If  ObjRed is Nothing Then Exit Sub
        StrValue = InputBox "Insert TimeOut", "Input Value"
        ObjRed.TimeOut = Val(StrValue)
        MsgBox "Retries -> " & ObjRed.TimeOut
End Sub
```

40. OPCAECmdTarget

Refresh, OPCAECmdTarget Function

Syntax Refresh()

Description This function is not currently supported.

Parameter	Description
None	None

Result Boolean

Reconnect, OPCAECmdTarget Function

Syntax Reconnect()

Description This function is not currently supported.

Parameter	Description
None	None

Result Boolean

GetXMLSettings, OPCAECmdTarget Function

Syntax GetXMLSettings()

Description This function is not currently supported.

Parameter	Description
None	None

Result String

GetServerVendorInfo, OPCAECmdTarget Function

Syntax GetServerVendorInfo()

Description This function is not currently supported.

Parameter	Description
None	None

Result String

GetServerStatus, OPCAECmdTarget Function

Syntax GetServerStatus()

Description This function is not currently supported.

Parameter	Description
None	None

Result Long

GetNumObjectsInHeap, OPCAECmdTarget Function

Syntax GetNumObjectsInHeap()

Description This function is not currently supported.

Parameter	Description
None	None

Result Long

GetNumEventInQueue, OPCAECmdTarget Function

Syntax GetNumEventInQueue()

Description This function is not currently supported.

Parameter	Description
None	None

Result Long

GetEventInQueueAt, OPCAECmdTarget Function

Syntax GetEventInQueueAt(_nIndex)

Description Function not currently supported.

Parameter	Description
nIndex As Integer	Index corresponding to event

Result String

Server, OPCAECmdTarget Property

Syntax Server = _Long

Description This property is not currently supported.

Parameter	Description
None	None

Result String

ReconnectTime, OPCAECmdTarget Property

Syntax ReconnectTime = _Long

Description This property is not currently supported.

Parameter	Description
None	None

Result Long

Node, OPCAECmdTarget Property

Syntax Node = _Long

Description This property is not currently supported.

Parameter	Description
None	None

Result String

MaxEventQueue, OPCAECmdTarget Property

Syntax MaxEventQueue = _Long

Description This property is not currently supported.

Parameter	Description
None	None

Result Long

41. OPCClientCmdTarget

ClearDynOPCItemTypeCache, OPCClientCmdTarget Function

Syntax ClearDynOPCItemTypeCache

Description This function allows you to clear the cache memory for managing dynamic variables in-use.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim vResult As Long
        Set objOPC = GetOPCClient
        vResult = objOPC.ClearDynOPCItemTypeCache
        MsgBox "ClearDynOPCItemTypeCache = " & vResult,vbInformation,GetProjectTitle
        Set objOPC = Nothing
End Sub
```

GetOPCAEClientObject, OPCClientCmdTarget Function

Syntax GetOPCAEClientObject(_lpszServerName)

Description This function returns a OPAECmdTarget object type though which you can manage the properties and methods relating to the OPC Alarm Event communication standard.

This functino is not supported in Windows CE.(if used always returns 'null')

Parameter	Description
lpszServerName As String	Registration name of OPC server.

Result Object
If Function has been executed successfully it will retrieve an object of type OPCAECmdTarget if otherwise Nothing is returned.

Example:

```
Public Sub Click()
```

```
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCAE As OPCAECmdTarget
        Dim vResult As Long
        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCAE = objOPC.GetOPCAEClientObject("Softing OPC Toolbox Demo OPC AE
        Server")
        If objOPCAE Is Nothing Then Exit Sub
        MsgBox "OPCAE_XMLSettings are: " &
        objOPCAE.GetXMLSettings,vbInformation,GetProjectTitle
        Set objOPC = Nothing
End Sub
```

GetOPCDAClientObject, OPCClientCmdTarget Function

Syntax GetOPCDAClientObject(_lpszServerName)

Description This function returns a OPCClientObjCmdTargetat object type through which you can manage the properties and methods relating to OPC Data Access communication standard.

Parameter	Description
lpszServerName As String	Registration name of the OPC Server

Result Object
If Function has been executed successfully it will retrieve an object of type OPCClientObjCmdTarget if otherwise Nothing is returned.

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget
        Dim objOPCAE As OPCAECmdTarget
        Dim vResult As Long

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub

        Set objOPCAE = objOPC.GetOPCAEClientObject("Softing OPC Toolbox Demo OPC DA
        Server")
        If Not objOPCAE Is Nothing Then
                MsgBox "objOPCAE_XMLSettings are: " &
                objOPCAE.GetXMLSettings,vbInformation,GetProjectTitle
                Set objOPCAE = Nothing
        End If
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC DA
        Server")
        If objOPCDA Is Nothing Then Exit Sub
        MsgBox "OPCDA_XMLSettings are: " &
        objOPCDA.GetXMLSettings,vbInformation,GetProjectTitle

        Set objOPCDA = Nothing
        Set objOPC = Nothing
End Sub
```

GetXMLSettings, OPCClientCmdTarget Function

Syntax GetXMLSettings

Description This function returns the OPCClient's definition string in XML format.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim vResult As String
        Set objOPC = GetOPCClient
        vResult = objOPC.GetXMLSettings
        MsgBox "XMLSettings are:  " & vResult,vbInformation,GetProjectTitle
        Set objOPC = Nothing
End Sub
```

ReconnectAll, OPCClientCmdTarget Function

Syntax ReconnectAll(_bWait)

Description This function enables or disables the how long the Client must wait before reconnecting to the Server when communication is lost.

Parameter	Description
bWait As Boolean	Enables how long Client must wait before reconnecting to server.

Result Boolean

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim bResult As Boolean
        Set objOPC = GetOPCClient
        vResult = objOPC.ReconnectAll(True)
        MsgBox "ReconnectAll =  " & bResult,vbInformation,GetProjectTitle
        Set objOPC = Nothing
End Sub
```

PoolOPCClientTest, OPCClientCmdTarget Property

Syntax PoolOPCClientTest = _Long

Description This property sets or returns the number of Threads to be used for the OPC Items test process when connected to the Server.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        MsgBox "PoolOPCClientTest = " &
        objOPC.PoolOPCClientTest,vbInformation,GetProjectTitle
        Set objOPC = Nothing
End Sub
```

RefreshRateDynamicOPC, OPCClientCmdTarget Property

Syntax RefreshRateDynamicOPC = _Long

Description This property sets or returns the Refresh Rate time for the project's dynamic OPC Items.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        MsgBox "RefreshRateDynamicOPC = " &
        objOPC.RefreshRateDynamicOPC,vbInformation,GetProjectTitle
        Set objOPC = Nothing
End Sub
```

StartupTimeout, OPCClientCmdTarget Property

Syntax StartupTimeout = _Long

Description This property sets or returns the timeout, in milliseconds, for the OPC Item's initializing process upon connecting to the Server.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        MsgBox " StartupTimeout = " & objOPC.
        StartupTimeout,vbInformation,GetProjectTitle
        Set objOPC = Nothing
End Sub
```

TimeoutDynamicOperation, OPCClientCmdTarget Property

Syntax TimeoutDynamicOperation = _Long

Description This property sets or returns the Timeout, expressed in milliseconds, for the operation of inserting a new dynamic OPC Item.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        MsgBox "TimeoutDynamicOperation = " &
        objOPC.TimeoutDynamicOperation,vbInformation,GetProjectTitle
        Set objOPC = Nothing
End Sub
```

TimeoutOPCClientTest, OPCClientCmdTarget Property

Syntax TimeoutOPCClientTest = _Long

Description This property sets or returns the Timeout time expresses in milliseconds for testing an OPC Item when connecting to the Server.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        MsgBox "TimeoutOPCClientTest= " &
        objOPC.TimeoutOPCClientTest,vbInformation,GetProjectTitle
        Set objOPC = Nothing
End Sub
```

42. OPCClientGroupObjCmdTarget

UpdateGroupProperties, OPCClientGroupObjCmdTarget Function

Syntax UpdateGroupProperties()

Description The function forces the property update of the group of OPC items.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget
        Dim GroupOne As OPCClientGroupObjCmdTarget

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC
        DA Server")
        If objOPCDA Is Nothing Then Exit Sub
        Set GroupOne = objOPCDA.GetGroupObject("Group one")
        If GroupOne Is Nothing Then Exit Sub
        GroupOne.DeadBand = 0.50
        GroupOne.UpdateGroupProperties

        Set objOPCDA = Nothing
        Set objOPC = Nothing
        Set GroupOne = Nothing
End Sub
```

IsGroupConnected, OPCClientGroupObjCmdTarget Function

Syntax IsGroupConnected()

Description This function returns the True boolean value when the Items Group is connected.

Parameter	Description

None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget
        Dim GroupOne As OPCClientGroupObjCmdTarget

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC
        DA Server")
        If objOPCDA Is Nothing Then Exit Sub
        Set GroupOne = objOPCDA.GetGroupObject("Group one")
        If GroupOne Is Nothing Then Exit Sub
        If GroupOne.IsGroupConnected Then
                MsgBox "GroupOne is connected",vbInformation,GetProjectTitle
        Else
                MsgBox "GroupOne is NOT connected",vbInformation,GetProjectTitle
        End If
        Set objOPCDA = Nothing
        Set objOPC = Nothing
        Set GroupOne = Nothing
End Sub
```

GetXMLSettings, OPCClientGroupObjCmdTarget Function

Syntax GetXMLSettings()

Description This function returns the XML settings and configuration text of the OPC group referenced and inserted in the project. This is actually the contents of the "projectname.movopcclient" resource file relating to the referenced group.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget
        Dim GroupOne As OPCClientGroupObjCmdTarget

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC
        DA Server")
        If objOPCDA Is Nothing Then Exit Sub
        Set GroupOne = objOPCDA.GetGroupObject("Group one")
        If GroupOne Is Nothing Then Exit Sub
```

```
        MsgBox "XMLSettings are " & GroupOne.GetXMLSettings,vbInformation,GetProjectTitle

        Set objOPCDA = Nothing
        Set objOPC = Nothing
        Set GroupOne = Nothing
End Sub
```

GetServerObject, OPCClientGroupObjCmdTarget Function

Syntax GetServerObject()

Description This function returns the OPCClientObjCmdTarget object relating to the server belonging to the referenced Group.

Parameter	Description
None	None

Result Object
If Function has been executed successfully it will retrieve an object of type OPCClientObjCmdTarget if otherwise Nothing is returned.

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget
        Dim GroupOne As OPCClientGroupObjCmdTarget

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC DA Server")
        If objOPCDA Is Nothing Then Exit Sub
        Set GroupOne = objOPCDA.GetGroupObject("Group one")
        If GroupOne Is Nothing Then Exit Sub
        GroupOne.GetServerObject.ConnectServer(True)
        Set objOPCDA = Nothing
        Set objOPC = Nothing
        Set GroupOne = Nothing
End Sub
```

GetNumObjectsInHeap, OPCClientGroupObjCmdTarget Function

Syntax GetNumObjectsInHeap

Description This function returns the number of objects loaded in memory.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget
        Dim GroupOne As OPCClientGroupObjCmdTarget

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC
        DA Server")
        If objOPCDA Is Nothing Then Exit Sub
        Set GroupOne = objOPCDA.GetGroupObject("Group one")
        If GroupOne Is Nothing Then Exit Sub
        MsgBox "NumObjectsInHeap are " &
        GroupOne.GetNumObjectsInHeap,vbInformation,GetProjectTitle
        Set objOPCDA = Nothing
        Set objOPC = Nothing
        Set GroupOne = Nothing
End Sub
```

GetName, OPCClientGroupObjCmdTarget Function

Syntax GetName()

Description This function returns the name of the reference Group.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget
        Dim GroupOne As OPCClientGroupObjCmdTarget
        Dim ItemOne As OPCClientItemObjCmdTarget

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC
        DA Server")
        If objOPCDA Is Nothing Then Exit Sub
        Set GroupOne = objOPCDA.GetGroupObject("Group one")
        If GroupOne Is Nothing Then Exit Sub
                MsgBox "GroupOne Name is" &
                GroupOne.GetName,vbInformation,GetProjectTitle
        Set ItemOne  = Nothing
        Set objOPCDA = Nothing
        Set objOPC = Nothing
        Set GroupOne = Nothing
End Sub
```

GetItemObject, OPCClientGroupObjCmdTarget Function

Syntax GetItemObject(_lpszItemName)

Description This function permits you to retrieve the OPCClientItemObjCmdTarget object relating to the reference item.

Parameter	Description
lpszItemName As String	Name of item to use for getting the object

Result Object
If Function has been executed successfully it will retrieve an object of type OPCClientItemObjCmdTarget if otherwise Nothing is returned.

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget
        Dim GroupOne As OPCClientGroupObjCmdTarget
        Dim ItemOne As OPCClientItemObjCmdTarget

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC
        DA Server")
        If objOPCDA Is Nothing Then Exit Sub
        Set GroupOne = objOPCDA.GetGroupObject("Group one")
        If GroupOne Is Nothing Then Exit Sub
        Set ItemOne  = GroupOne.GetItemObject("Var00001")
        If ItemOne  Is Nothing Then Exit Sub
        MsgBox "ItemOne  ItemID is" & ItemOne.ItemID,vbInformation,GetProjectTitle
        Set ItemOne   = Nothing
        Set objOPCDA = Nothing
        Set objOPC = Nothing
        Set GroupOne = Nothing
End Sub
```

ConnectGroup, OPCClientGroupObjCmdTarget Function

Syntax ConnectGroup(_bWait)

Description This function allows you to force the connection relating to the reference Group's Items. The boolean parameter specifies whether Movicon must attend the reconnection waiting time before executing the connection.

Parameter	Description
bWait As Boolean	Enables the reconnection waiting time.

Result Boolean

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget
        Dim GroupOne As OPCClientGroupObjCmdTarget

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC
        DA Server")
        If objOPCDA Is Nothing Then Exit Sub
        Set GroupOne = objOPCDA.GetGroupObject("Group one")
        GroupOne.ConnectGroup(False)
        Set objOPCDA = Nothing
        Set objOPC = Nothing
        Set GroupOne = Nothing
End Sub
```

UpdateRate, OPCClientGroupObjCmdTarget Property

Syntax UpdateRate = _Long

Description This property allows you set the Update rate in milliseconds of the Group's Items.

This update time is a parameter which is passed to the Server. In cases where the update time is too low, the Server will use its minimum value.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget
        Dim GroupOne As OPCClientGroupObjCmdTarget

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC
        DA Server")
        If objOPCDA Is Nothing Then Exit Sub
        Set GroupOne = objOPCDA.GetGroupObject("Group one")
        MsgBox "UpdateRate is " &
        GroupOne.UpdateRate,vbInformation,GetProjectTitle
        Set objOPCDA = Nothing
        Set objOPC = Nothing
        Set GroupOne = Nothing
End Sub
```

TimeBias, OPCClientGroupObjCmdTarget Property

Syntax TimeBias = _Long

Description This property allows you to set a value in minutes which will be used to convert the Time Stamp property of the Items contained in the Group to the device's local time. Normally it is not necessary to change this setting from the default zero value.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget
        Dim GroupOne As OPCClientGroupObjCmdTarget

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC
        DA Server")
        If objOPCDA Is Nothing Then Exit Sub
        Set GroupOne = objOPCDA.GetGroupObject("Group one")
        MsgBox "TimeBias is " & GroupOne.TimeBias,vbInformation,GetProjectTitle
        Set objOPCDA = Nothing
        Set objOPC = Nothing
        Set GroupOne = Nothing
End Sub
```

NotInUseRefreshRate, OPCClientGroupObjCmdTarget Property

Syntax NotInUseRefreshRate = _Long

Description This property permits you to set the Refresh rate in milliseconds of the Group's items when the variables associated to them are not in use in the project.

This Refresh time must be set higher than the "refresh Every" time.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget
        Dim GroupOne As OPCClientGroupObjCmdTarget

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC
        DA Server")
        If objOPCDA Is Nothing Then Exit Sub
        Set GroupOne = objOPCDA.GetGroupObject("Group one")
        MsgBox "NotInUseRefreshRate is " & GroupOne.NotInUseRefreshRate
        ,vbInformation,GetProjectTitle
        Set objOPCDA = Nothing
        Set objOPC = Nothing
        Set GroupOne = Nothing
End Sub
```

NotInUseDisableGroup, OPCClientGroupObjCmdTarget Property

Syntax NotInUseDisableGroup = _Boolean

Description This property allows you to deactivate the Group when the variables associated to the Items are not in use.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget
        Dim GroupOne As OPCClientGroupObjCmdTarget

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC
        DA Server")
        If objOPCDA Is Nothing Then Exit Sub
        Set GroupOne = objOPCDA.GetGroupObject("Group one")
        MsgBox "NotInUseDisableGroup is " & GroupOne.NotInUseDisableGroup
        ,vbInformation,GetProjectTitle
        Set objOPCDA = Nothing
        Set objOPC = Nothing
        Set GroupOne = Nothing
End Sub
```

LocalID, OPCClientGroupObjCmdTarget Property

Syntax LocalID = _Long

Description This property returns the LocalID relating to the referenced group.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget
        Dim GroupOne As OPCClientGroupObjCmdTarget

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC
        DA Server")
        If objOPCDA Is Nothing Then Exit Sub
        Set GroupOne = objOPCDA.GetGroupObject("Group one")
        MsgBox "LocalID is " & GroupOne.LocalID,vbInformation,GetProjectTitle
        Set objOPCDA = Nothing
        Set objOPC = Nothing
        Set GroupOne = Nothing
End Sub
```

EnableGroup, OPCClientGroupObjCmdTarget Property

Syntax EnableGroup = _Boolean

Description This property enables or disables the reference Group and thus the Items contained within.
You must make sure that the variables in-use management is enabled, each time a variable from the group goes back into use the group will automatically reactivate.

the "EnableGroup" function interrogates the opc server to ask whether the OPC group is active in its workspace. When the "Deactivate not in use" option has been checked in the Movicon 11 group, the "EnableGroup" function may return always 'true' even after having executed "EnableGroup=false", because the group is still active in the opc server (simply due to changed refresh time).

Warning! the "EnableGroup = Not EnableGroup" syntax (i.e. inserted in a button) always returns 'EnableGroup=False'; to get property denial you will need to insert, for instance, 2 buttons in which in which you should insert: 'EnableGroup=true' in the first and 'EnableGroup=false' in the second.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget
        Dim GroupOne As OPCClientGroupObjCmdTarget

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC
        DA Server")
        If objOPCDA Is Nothing Then Exit Sub
        Set GroupOne = objOPCDA.GetGroupObject("Group one")
        If GroupOne.EnableGroupthen
                MsgBox "GroupOne is enabled",vbInformation,GetProjectTitle
        Else
                MsgBox "GroupOne is NOT enabled",vbInformation,GetProjectTitle
        End If
        Set objOPCDA = Nothing
        Set objOPC = Nothing
        Set GroupOne = Nothing
End Sub
```

DeadBand, OPCClientGroupObjCmdTarget Property

Syntax DeadBand = _Single

Description This property sets or returns a Dead Band percentage from 0 to 100 for the Items contained in the referenced grouped. The dead band is applied to Group Items which have EU Type parameters set on Analog, in this way the EU Low and EU High parameters are used for calculating the Item's interval. The interval is multiplied by the dead band to generate an exception limit when the difference between the previously read value and the new value exceed this calculation.
The dead band is used for eliminating problems caused by disturbance in reading analogic values: in exceptional cases, the Item remains at the previous value read.

Parameter	Description
None	None

Result Single

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget
```

```
        Dim GroupOne As OPCClientGroupObjCmdTarget

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC
        DA Server")
        If objOPCDA Is Nothing Then Exit Sub
        Set GroupOne = objOPCDA.GetGroupObject("Group one")
        MsgBox "DeadBand is " & GroupOne.DeadBand ,vbInformation,GetProjectTitle
        Set objOPCDA = Nothing
        Set objOPC = Nothing
        Set GroupOne = Nothing
End Sub
```

Active, OPCClientGroupObjCmdTarget Property

Syntax Active = _Boolean

Description This property is read only and returns the "Active" option license set in development mode. By using the "EnableGroup" groups can be enabled/disabled in runtime. However, care must be taken because when the variables in use management is enabled, each time a variable from the group goes into use the group is automatically reactivated.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget
        Dim GroupOne As OPCClientGroupObjCmdTarget

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC
        DA Server")
        If objOPCDA Is Nothing Then Exit Sub
        Set GroupOne = objOPCDA.GetGroupObject("Group one")
        If GroupOne.Active then
                MsgBox "GroupOne is active",vbInformation,GetProjectTitle
        Else
                MsgBox "GroupOne is NOT active",vbInformation,GetProjectTitle
        End If
        Set objOPCDA = Nothing
        Set objOPC = Nothing
        Set GroupOne = Nothing
End Sub
```

43. OPCClientItemObjCmdTarget

ForceReadData, OPCClientItemObjCmdTarget Function

Syntax ForceReadData(_bFromDevice)

Description This function force reads the referenced Item.

Parameter	Description
bFromDevice as boolean	The bFromDevice parameter specifies to Movicon to read data from the field when set with the true boolean value.

Result Boolean

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget
        Dim GroupOne As OPCClientGroupObjCmdTarget
        Dim ItemOne As OPCClientItemObjCmdTarget

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC
        DA Server")
        If objOPCDA Is Nothing Then Exit Sub
        Set GroupOne = objOPCDA.GetGroupObject("Group one")
        If GroupOne Is Nothing Then Exit Sub
        Set ItemOne  = GroupOne.GetItemObject("Var00001")
        If ItemOne  Is Nothing Then Exit Sub
        ItemOne.ForceReadData(True)
        Set ItemOne   = Nothing
        Set objOPCDA = Nothing
        Set objOPC = Nothing
        Set GroupOne = Nothing
End Sub
```

ForceWriteData, OPCClientItemObjCmdTarget Function

Syntax ForceWriteData = Boolean

Description This method consents the item to be written, even though the linked variable value had not changed. This method is handy when using OPC Servers that interface with LON or EIB networks, where it may sometimes be necessary to re-send the same value even if it hasn't changed.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget
        Dim GroupOne As OPCClientGroupObjCmdTarget
        Dim ItemOne As OPCClientItemObjCmdTarget
        Dim i As Integer

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC
        DA Server")
        If objOPCDA Is Nothing Then Exit Sub
        Set GroupOne = objOPCDA.GetGroupObject("Group one")
        If GroupOne Is Nothing Then Exit Sub
        Set ItemOne  = GroupOne.GetItemObject("Var00001")
        If ItemOne  Is Nothing Then Exit Sub
        Dim i As Integerv
            MsgBox "SyncDataAtStartup is " &
            CBool(ItemOne.ForceWriteData),vbInformation, GetProjectTitle
        Next i
        Set ItemOne = Nothing
        Set objOPCDA = Nothing
        Set objOPC = Nothing
        Set GroupOne = Nothing
End Sub
```

GetGroupObject, OPCClientItemObjCmdTarget Function

Syntax GetGroupObject()

Description This function returns a OPCClientGroupObjCmdTarget type object for managing methods and properties of the group belonging to the reference item.

Parameter	Description
None	None

Result Object
If Function has been executed successfully it will retrieve an object of type OPCClientGroupObjCmdTarget if otherwise Nothing is returned.

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget
        Dim GroupOne As OPCClientGroupObjCmdTarget
        Dim ItemOne As OPCClientItemObjCmdTarget
```

```
        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC
        DA Server")
        If objOPCDA Is Nothing Then Exit Sub
        Set GroupOne = objOPCDA.GetGroupObject("Group one")
        If GroupOne Is Nothing Then Exit Sub
        Set ItemOne  = GroupOne.GetItemObject("Var00001")
        If ItemOne  Is Nothing Then Exit Sub
        ItemOne.GetGroupObject.DeadBand = 0.25
        ItemOne.GetGroupObject.UpdateGroupProperties
        Set ItemOne   = Nothing
        Set objOPCDA = Nothing
        Set objOPC = Nothing
        Set GroupOne = Nothing
End Sub
```

GetItemQuality, OPCClientItemObjCmdTarget Function

Syntax GetItemQuality()

Description This function returns the quality of the referenced Item.

The returned values are 'quality' values such as detailed for the OPC:

252 = OPC_STATUS_MASK
3 = OPC_LIMIT_MASK
0 = OPC_QUALITY_BAD
64 = OPC_QUALITY_UNCERTAIN
192 = OPC_QUALITY_GOOD
4 = OPC_QUALITY_CONFIG_ERROR
8 = OPC_QUALITY_NOT_CONNECTED
12 = OPC_QUALITY_DEVICE_FAILURE
16 = OPC_QUALITY_SENSOR_FAILURE
20 = OPC_QUALITY_LAST_KNOWN
24 = OPC_QUALITY_COMM_FAILURE
28 = OPC_QUALITY_OUT_OF_SERVICE
68 = OPC_QUALITY_LAST_USABLE
80 = OPC_QUALITY_SENSOR_CAL
84 = OPC_QUALITY_EGU_EXCEEDED
88 = OPC_QUALITY_SUB_NORMAL
216 = OPC_QUALITY_LOCAL_OVERRIDE

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget
        Dim GroupOne As OPCClientGroupObjCmdTarget
        Dim ItemOne As OPCClientItemObjCmdTarget

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
```

```
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC
        DA Server")
        If objOPCDA Is Nothing Then Exit Sub
        Set GroupOne = objOPCDA.GetGroupObject("Group one")
        If GroupOne Is Nothing Then Exit Sub
        Set ItemOne  = GroupOne.GetItemObject("Var00001")
        If ItemOne  Is Nothing Then Exit Sub
        MsgBox "ItemOne Quality is " &
        ItemOne.GetItemQuality,vbInformation,GetProjectTitle
        Set ItemOne  = Nothing
        Set objOPCDA = Nothing
        Set objOPC = Nothing
        Set GroupOne = Nothing
End Sub
```

GetItemTimeStamp, OPCClientItemObjCmdTarget Function

Syntax GetItemTimeStamp()

Description This function returns the last Time Stamp value relating to the referenced item.

Parameter	Description
None	None

Result Date

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget
        Dim GroupOne As OPCClientGroupObjCmdTarget
        Dim ItemOne As OPCClientItemObjCmdTarget

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC
        DA Server")
        If objOPCDA Is Nothing Then Exit Sub
        Set GroupOne = objOPCDA.GetGroupObject("Group one")
        If GroupOne Is Nothing Then Exit Sub
        Set ItemOne  = GroupOne.GetItemObject("Var00001")
        If ItemOne  Is Nothing Then Exit Sub
        MsgBox "Item TimeStamp is " &
        ItemOne.GetItemTimeStamp,vbInformation,GetProjectTitle
        Set ItemOne  = Nothing
        Set objOPCDA = Nothing
        Set objOPC = Nothing
        Set GroupOne = Nothing
End Sub
```

GetLinkedVariableObject, OPCClientItemObjCmdTarget Function

Syntax GetLinkedVariableObject()

Description This function returns the DBVarObjCmdTarget object relating to the Real Time DB variable of the project associated to the referenced OPC item.

Parameter	Description
None	None

Result Object
If Function has been executed successfully it will retrieve an object of type DBVarObjCmdTarget if otherwise Nothing is returned.

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget
        Dim GroupOne As OPCClientGroupObjCmdTarget
        Dim ItemOne As OPCClientItemObjCmdTarget

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC
        DA Server")
        If objOPCDA Is Nothing Then Exit Sub
        Set GroupOne = objOPCDA.GetGroupObject("Group one")
        If GroupOne Is Nothing Then Exit Sub
        Set ItemOne  = GroupOne.GetItemObject("Var00001")
        If ItemOne  Is Nothing Then Exit Sub
        MsgBox "Linked variable BGColorProp is " &
        ItemOne.GetLinkedVariableObject.BGColorProp ,vbInformation,GetProjectTitle
        Set ItemOne  = Nothing
        Set objOPCDA = Nothing
        Set objOPC = Nothing
        Set GroupOne = Nothing
End Sub
```

GetNumObjectsInHeap, OPCClientItemObjCmdTarget Function

Syntax GetNumObjectsInHeap()

Description This function returns the number of objects loaded in memory.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget
        Dim GroupOne As OPCClientGroupObjCmdTarget
        Dim ItemOne As OPCClientItemObjCmdTarget

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC
        DA Server")
        If objOPCDA Is Nothing Then Exit Sub
        Set GroupOne = objOPCDA.GetGroupObject("Group one")
        If GroupOne Is Nothing Then Exit Sub
        Set ItemOne  = GroupOne.GetItemObject("Var00001")
        If ItemOne  Is Nothing Then Exit Sub
        MsgBox "NumObjectsInHeap are " &
        ItemOne.GetNumObjectsInHeap,vbInformation,GetProjectTitle
        Set ItemOne  = Nothing
        Set objOPCDA = Nothing
        Set objOPC = Nothing
        Set GroupOne = Nothing
End Sub
```

GetXMLSettings, OPCClientItemObjCmdTarget Function

Syntax GetXMLSettings()

Description This function returns the settings and configuration XML text relating to the OPC item referenced and inserted in the project. This actually contains the "projectname.movopcclient" relating to the referenced item.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget
        Dim GroupOne As OPCClientGroupObjCmdTarget
        Dim ItemOne As OPCClientItemObjCmdTarget

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC
        DA Server")
        If objOPCDA Is Nothing Then Exit Sub
        Set GroupOne = objOPCDA.GetGroupObject("Group one")
        If GroupOne Is Nothing Then Exit Sub
        Set ItemOne  = GroupOne.GetItemObject("Var00001")
        If ItemOne  Is Nothing Then Exit Sub
        MsgBox "XML Settings are " &
        ItemOne.GetXMLSettings,vbInformation,GetProjectTitle
        Set ItemOne  = Nothing
        Set objOPCDA = Nothing
```

```
        Set objOPC = Nothing
        Set GroupOne = Nothing
End Sub
```

IsItemConnected, OPCClientItemObjCmdTarget Function

Syntax IsItemConnected()

Description This function returns the true boolean value when the referenced Item is connected.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget
        Dim GroupOne As OPCClientGroupObjCmdTarget
        Dim ItemOne As OPCClientItemObjCmdTarget

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC
        DA Server")
        If objOPCDA Is Nothing Then Exit Sub
        Set GroupOne = objOPCDA.GetGroupObject("Group one")
        If GroupOne Is Nothing Then Exit Sub
        Set ItemOne  = GroupOne.GetItemObject("Var00001")
        If ItemOne  Is Nothing Then Exit Sub
                MsgBox "Is " & CBool(ItemOne.IsItemConnected) & " that the item is
                connected",vbInformation,GetProjectTitle
        Set ItemOne   = Nothing
        Set objOPCDA = Nothing
        Set objOPC = Nothing
        Set GroupOne = Nothing
End Sub
```

ReconnectItem, OPCClientItemObjCmdTarget Function

Syntax ReconnectItem(_bWait)

Description This function allows you to force the reconnection to item by specifying whether the waitng time, set in the 'Reconnection Time' property, is necessary or not.

Parameter	Description

bWait As Boolean	Enables the reconnection waiting time

Result Boolean

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget
        Dim GroupOne As OPCClientGroupObjCmdTarget
        Dim ItemOne As OPCClientItemObjCmdTarget

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC
        DA Server")
        If objOPCDA Is Nothing Then Exit Sub
        Set GroupOne = objOPCDA.GetGroupObject("Group one")
        If GroupOne Is Nothing Then Exit Sub
        Set ItemOne  = GroupOne.GetItemObject("Var00001")
        If ItemOne  Is Nothing Then Exit Sub
        ItemOne.ReconnectItem(True)
        Set ItemOne   = Nothing
        Set objOPCDA = Nothing
        Set objOPC = Nothing
        Set GroupOne = Nothing
End Sub
```

EnableRead, OPCClientItemObjCmdTarget Property

Syntax EnableRead = _Boolean

Description This property, if set to the boolean true value, enables the referenced item in read. In this case any modifications to the item within the server will update the Movicon project's variable.

Parameter	Description
None	None

Result Boolean

Example:

```
        Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget
        Dim GroupOne As OPCClientGroupObjCmdTarget
        Dim ItemOne As OPCClientItemObjCmdTarget

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC
        DA Server")
        If objOPCDA Is Nothing Then Exit Sub
        Set GroupOne = objOPCDA.GetGroupObject("Group one")
```

```
        If GroupOne Is Nothing Then Exit Sub
        Set ItemOne  = GroupOne.GetItemObject("Var00001")
        If ItemOne  Is Nothing Then Exit Sub
        MsgBox "ItemOne.EnableRead is " & CBool(ItemOne.EnableRead)
        ,vbInformation,GetProjectTitle
        Set ItemOne  = Nothing
        Set objOPCDA = Nothing
        Set objOPC = Nothing
        Set GroupOne = Nothing
End Sub
```

EnableWrite, OPCClientItemObjCmdTarget Property

Syntax EnableWrite = _Boolean

Description This property, when set with the true boolean value, enables the reference items in write. In this case the item will write in the server when any modifications are made to the variable within the Movicon project.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget
        Dim GroupOne As OPCClientGroupObjCmdTarget
        Dim ItemOne As OPCClientItemObjCmdTarget

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC
        DA Server")
        If objOPCDA Is Nothing Then Exit Sub
        Set GroupOne = objOPCDA.GetGroupObject("Group one")
        If GroupOne Is Nothing Then Exit Sub
        Set ItemOne  = GroupOne.GetItemObject("Var00001")
        If ItemOne  Is Nothing Then Exit Sub
                MsgBox "ItemOne.EnableWrite is " & CBool(ItemOne.EnableWrite)
                ,vbInformation,GetProjectTitle
        Set ItemOne  = Nothing
        Set objOPCDA = Nothing
        Set objOPC = Nothing
        Set GroupOne = Nothing
End Sub
```

ItemID, OPCClientItemObjCmdTarget Property

Syntax ItemID = _String

Description This property returns the ID relating to the referenced item.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget
        Dim GroupOne As OPCClientGroupObjCmdTarget
        Dim ItemOne As OPCClientItemObjCmdTarget

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC
        DA Server")
        If objOPCDA Is Nothing Then Exit Sub
        Set GroupOne = objOPCDA.GetGroupObject("Group one")
        If GroupOne Is Nothing Then Exit Sub
        Set ItemOne  = GroupOne.GetItemObject("Var00001")
        If ItemOne  Is Nothing Then Exit Sub
                MsgBox "ItemID  is " & ItemOne.ItemID ,vbInformation,GetProjectTitle
        Set ItemOne   = Nothing
        Set objOPCDA = Nothing
        Set objOPC = Nothing
        Set GroupOne = Nothing
End Sub
```

ItemPath, OPCClientItemObjCmdTarget Property

Syntax ItemPath = _String

Description This property returns the OPC path relating to the referenced item.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget
        Dim GroupOne As OPCClientGroupObjCmdTarget
        Dim ItemOne As OPCClientItemObjCmdTarget

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC
        DA Server")
        If objOPCDA Is Nothing Then Exit Sub
        Set GroupOne = objOPCDA.GetGroupObject("Group one")
        If GroupOne Is Nothing Then Exit Sub
        Set ItemOne  = GroupOne.GetItemObject("Var00001")
        If ItemOne  Is Nothing Then Exit Sub
                MsgBox "ItemPath is " & ItemOne.ItemPath
                ,vbInformation,GetProjectTitle
        Set ItemOne  = Nothing
        Set objOPCDA = Nothing
        Set objOPC = Nothing
        Set GroupOne = Nothing
End Sub
```

LinkedVariable, OPCClientItemObjCmdTarget Property

Syntax LinkedVariable = _String

Description This property returns the name of the Real Time DB variable of the project to be associated to the Server's OPC item.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget
        Dim GroupOne As OPCClientGroupObjCmdTarget
        Dim ItemOne As OPCClientItemObjCmdTarget

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC
        DA Server")
        If objOPCDA Is Nothing Then Exit Sub
        Set GroupOne = objOPCDA.GetGroupObject("Group one")
        If GroupOne Is Nothing Then Exit Sub
        Set ItemOne  = GroupOne.GetItemObject("Var00001")
        If ItemOne  Is Nothing Then Exit Sub
                MsgBox "LinkedVariable is " & ItemOne.LinkedVariable
                ,vbInformation,GetProjectTitle
        Set ItemOne  = Nothing
        Set objOPCDA = Nothing
        Set objOPC = Nothing
        Set GroupOne = Nothing
```

```
End Sub
```

ReRead, OPCClientItemObjCmdTarget Property

Syntax ReRead = _Boolean

Description This property allows you to choose where to execute a synco read of the value each time a write is executed (sychronous or asynchronous). This may be necessary when using a OPC Server (see Rockwell's RsLinx) which does not manage asynchronous notifications properly.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget
        Dim GroupOne As OPCClientGroupObjCmdTarget
        Dim ItemOne As OPCClientItemObjCmdTarget

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC
        DA Server")
        If objOPCDA Is Nothing Then Exit Sub
        Set GroupOne = objOPCDA.GetGroupObject("Group one")
        If GroupOne Is Nothing Then Exit Sub
        Set ItemOne  = GroupOne.GetItemObject("Var00001")
        If ItemOne  Is Nothing Then Exit Sub
                MsgBox "SyncDataAtStartup is " & CBoole(ItemOne.ReRead)
                ,vbInformation,GetProjectTitle
        Set ItemOne  = Nothing
        Set objOPCDA = Nothing
        Set objOPC = Nothing
        Set GroupOne = Nothing
End Sub
```

SyncDataAtStartup, OPCClientItemObjCmdTarget Property

Syntax SyncDataAtStartup = _Boolean

Description This property allows you to choose whether to read the item at project startup, after it has been created, fo synchronize the variable's value with that from the field. This function is disabled for default to avoid using up too much time initializing the OPC communication due to all the items being set with sync. This property can be enabled in the items where it is absolutely necessary to synchronize the values.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget
        Dim GroupOne As OPCClientGroupObjCmdTarget
        Dim ItemOne As OPCClientItemObjCmdTarget

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC
        DA Server")
        If objOPCDA Is Nothing Then Exit Sub
        Set GroupOne = objOPCDA.GetGroupObject("Group one")
        If GroupOne Is Nothing Then Exit Sub
        Set ItemOne  = GroupOne.GetItemObject("Var00001")
        If ItemOne  Is Nothing Then Exit Sub
                MsgBox "SyncDataAtStartup is " &
                CBoole(ItemOne.SyncDataAtStartup) ,vbInformation,GetProjectTitle
        Set ItemOne  = Nothing
        Set objOPCDA = Nothing
        Set objOPC = Nothing
        Set GroupOne = Nothing
End Sub
```

vtType, OPCClientItemObjCmdTarget Property

Syntax vtType = _Integer

Description This property allows you to set the variable type in read from the Server even though it is advised to leave it in its original format.
The options are:

- Default Server: when this option box is enabled, no conversion of any type will take place following the read or write of an item. In this case it is advised to use a Movicon variable of the same type set in the Item's properties.
    - 0 is the value associated to this option
- Short, Long, Float,Double,etc.: According to the format set, the corresponding conversion will be executed in the item, therefore the project's assigned variable, will contain the data in the format indicated here:
    - short
    - long
    - float
    - double
    - string
    - boolean
    - char
    - byte

- word
- dword

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget
        Dim GroupOne As OPCClientGroupObjCmdTarget
        Dim ItemOne As OPCClientItemObjCmdTarget

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC
        DA Server")
        If objOPCDA Is Nothing Then Exit Sub
        Set GroupOne = objOPCDA.GetGroupObject("Group one")
        If GroupOne Is Nothing Then Exit Sub
        Set ItemOne  = GroupOne.GetItemObject("Var00001")
        If ItemOne  Is Nothing Then Exit Sub
                MsgBox "vtType is " & ItemOne.vtType ,vbInformation,GetProjectTitle
        Set ItemOne   = Nothing
        Set objOPCDA = Nothing
        Set objOPC = Nothing
        Set GroupOne = Nothing
End Sub
```

WriteSync, OPCClientItemObjCmdTarget Property

Syntax WriteSync = _Boolean

Description This property allows you to choose whether to execute a sync (default) or async write. The async writes are executed much faster than sync writes, but the OPC Server must be setup to receive high numbers of requests, for instance, when the variables change values continuously in the Movicon project. Fro this reason the default value is set at Sync which is slower but more reliable (works for all OPC Servers).

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget
        Dim GroupOne As OPCClientGroupObjCmdTarget
```

```
        Dim ItemOne As OPCClientItemObjCmdTarget

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC
        DA Server")
        If objOPCDA Is Nothing Then Exit Sub
        Set GroupOne = objOPCDA.GetGroupObject("Group one")
        If GroupOne Is Nothing Then Exit Sub
        Set ItemOne  = GroupOne.GetItemObject("Var00001")
        If ItemOne  Is Nothing Then Exit Sub
                MsgBox "SyncDataAtStartup is " & CBoole(ItemOne.WriteSync)
                ,vbInformation,GetProjectTitle
        Set ItemOne   = Nothing
        Set objOPCDA = Nothing
        Set objOPC = Nothing
        Set GroupOne = Nothing
End Sub
```

44. OPCClientObjCmdTarget

IsConnected, OPCClientObjCmdTarget Function

Syntax IsConnected()

Description This function returns the true boolean value when the server is connected.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC
        DA Server")
        If objOPCDA Is Nothing Then Exit Sub
        MsgBox "Server 'IsConnected' is " &
        objOPCDA.IsConnected,vbInformation,GetProjectTitle
        Set objOPCDA = Nothing
        Set objOPC = Nothing
End Sub
```

GetXMLSettings, OPCClientObjCmdTarget Function

Syntax GetXMLSettings()

Description This function returns the settings and configuration XML text relating to the OPC item referenced and inserted in the project. This actually contains the "projectname.movopcclient" relating to the referenced item.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC
        DA Server")
        If objOPCDA Is Nothing Then Exit Sub
        MsgBox "XMLSettings are " &
        objOPCDA.GetXMLSettings,vbInformation,GetProjectTitle
        Set objOPCDA = Nothing
        Set objOPC = Nothing
End Sub
```

GetServerVendorInfo, OPCClientObjCmdTarget Function

Syntax GetServerVendorInfo()

Description This function returns information relating to the referenced OPC DA server.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC
        DA Server")
        If objOPCDA Is Nothing Then Exit Sub
        MsgBox "Server VendorInfo are " &
        objOPCDA.GetServerVendorInfo,vbInformation,GetProjectTitle
        Set objOPCDA = Nothing
        Set objOPC = Nothing
End Sub
```

GetServerStatus, OPCClientObjCmdTarget Function

Syntax GetServerStatus()

Description This function returns the status of the referenced OPC DA server.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC
        DA Server")
        If objOPCDA Is Nothing Then Exit Sub
        MsgBox "Server Status is " &
        objOPCDA.GetServerStatus,vbInformation,GetProjectTitle
        Set objOPCDA = Nothing
        Set objOPC = Nothing
End Sub
```

GetServerName, OPCClientObjCmdTarget Function

Syntax GetServerName()

Description This function returns the name of the referenced OPC Da server.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC
        DA Server")
        If objOPCDA Is Nothing Then Exit Sub
        MsgBox "Server Name is " &
        objOPCDA.GetServerName,vbInformation,GetProjectTitle
        Set objOPCDA = Nothing
        Set objOPC = Nothing
End Sub
```

GetServerCLSID, OPCClientObjCmdTarget Function

Syntax GetServerCLSID()

Description This function returns the CLS ID relating to the referenced OPA DA server.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC
        DA Server")
        If objOPCDA Is Nothing Then Exit Sub
        MsgBox "Server CLS ID is " &
        objOPCDA.GetServerCLSID,vbInformation,GetProjectTitle
        Set objOPCDA = Nothing
        Set objOPC = Nothing
End Sub
```

GetOPCClientDocObj, OPCClientObjCmdTarget Function

Syntax GetOPCClientDocObj()

Description This function returns the Doc object relating to the referenced OPC server.

Parameter	Description
None	None

Result Object

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget
        Dim vResult As Long
        Dim DOCobj As Object

        Set objOPC = GetOPCClient
```

```
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing.OPCToolboxDemo_ServerDA.1")
        If objOPCDA Is Nothing Then Exit Sub
        Set DOCobj = objOPCDA.GetOPCClientDocObj
        If Not DOCobj Is Nothing Then
            ...
        End If
        Set DOCobj = Nothing
        Set objOPCDA = Nothing
        Set objOPC = Nothing
End Sub
```

GetNumObjectsInHeap, OPCClientObjCmdTarget Function

Syntax GetNumObjectsInHeap()

Description This function returns the number of object loaded in memory.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC DA Server")
        If objOPCDA Is Nothing Then Exit Sub
        MsgBox "NumObjectsInHeap are " & objOPCDA.GetNumObjectsInHeap,vbInformation,GetProjectTitle
        Set objOPCDA = Nothing
        Set objOPC = Nothing
End Sub
```

GetNodeName, OPCClientObjCmdTarget Function

Syntax GetNodeName()

Description This function returns the name of the OPC node relating to the referenced OPC DA server.

Parameter	Description

None	None

Result String

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC
        DA Server")
        If objOPCDA Is Nothing Then Exit Sub
        MsgBox "objOPCDA Node Name is " &
        objOPCDA.GetNodeName,vbInformation,GetProjectTitle
        Set objOPCDA = Nothing
        Set objOPC = Nothing
End Sub
```

GetGroupObject, OPCClientObjCmdTarget Function

Syntax GetGroupObject(_lpszGroupName)

Description This function allows you to get a OPCClientGroupObjCmdTarget object type for managing the properties and methods relating to the groups of reference Items in the OPC Client.

Parameter	Description
lpszGroupName As String	Name of group to be retrieved.

Result Object
If Function has been executed successfully it will retrieve an object of type OPCClientGroupObjCmdTarget if otherwise Nothing is returned.

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget
        Dim Gruppo1 As OPCClientGroupObjCmdTarget

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC DA
        Server")
        If objOPCDA Is Nothing Then Exit Sub
        Set Gruppo1 = objOPCDA.GetGroupObject("Gruppo uno")
        MsgBox "Gruppo1.Active is " & Gruppo1.Active,vbInformation,GetProjectTitle
        Set objOPCDA = Nothing
        Set objOPC = Nothing
        Set Gruppo1 = Nothing
End Sub
```

ConnectServer, OPCClientObjCmdTarget Function

Syntax ClearDynConnectServer(_bWait)

Description This function allows you to force the connection to the Server by specifying whether or not to enter how long the connection should wait before reconnecting in the 'Reconnect Time' property.

Parameter	Description
bWait As Boolean	To enable how long the connect should wait before reconnecting to server

Result Boolean

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget
        Set objOPC = GetOPCClient

        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC DA
        Server")
        If objOPCDA Is Nothing Then Exit Sub
        objOPCDA.ConnectServer(True)
        Set objOPCDA = Nothing
        Set objOPC = Nothing
End Sub
```

ReconnectTime, OPCClientObjCmdTarget Property

Syntax ReconnectTime = _Long

Description This property sets or returns the time, in millieseconds, after which the Server will be connected.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim objOPC As OPCClientCmdTarget
        Dim objOPCDA As OPCClientObjCmdTarget

        Set objOPC = GetOPCClient
        If objOPC Is Nothing Then Exit Sub
        Set objOPCDA = objOPC.GetOPCDAClientObject("Softing OPC Toolbox Demo OPC
        DA Server")
```

```
        If objOPCDA Is Nothing Then Exit Sub
        MsgBox "Server ReconnectTime is " & objOPCDA.ReconnectTime,vbInformation,GetProjectTitle
        Set objOPCDA = Nothing
        Set objOPC = Nothing
End Sub
```

ReReadDynamicItems, OPCClientObjCmdTarget Property

Syntax ReReadDynamicItems = _Bool

Description This property, which is applied only to items created in dynamic mode by being inserting the OPC link in a variable's "I/O fixed Address" property, is used for choosing whether to execute a synchronized value read each time a write is executed (synchro. or asynchro.). It may be necessary when using some OPC Servers (see Rockwell's RsLinx) that do not manage asynchronized notifications correctly.
That property can be both read and written, but its modification in runtime will not be applied to already dynamically created items.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim objOPCClient As OPCClientObjCmdTarge

        Set objOPCClient = GetOPCClient().GetOPCDAClientObject("Softing OPC Toolbox Demo OPC DA Server")
        If objOPCClient Is Nothing Then Exit Sub
        MsgBox "ReReadDynamicItems is " & objOPCClient.ReReadDynamicItems,vbInformation,GetProjectTitle
        Set objOPCClient = Nothing
End Sub
```

45. OPCServerCmdTarget

FireAEEvent, OPCServerCmdTarget Function

Syntax FireAEEvent(_lpszSource, _lpszMessage, _dwPriority, _dwSeverity)

Description This function is not managed. The value will always return "True".

Parameter	Description
None	None

Result Boolean

Example:

GetNumServingTags, OPCServerCmdTarget Function

Syntax GetNumServingTags()

Description This function returns the number of tags that the project's Movicon OPC Server published towards the OPC Clients.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim objOPCServe As OPCServerCmdTarget
        Dim sOPCNumServingTags As Long

        Set objOPCServe = GetOPCServer()
        If Not objOPCServe Is Nothing Then
                sOPCNumServingTags = objOPCServe.GetNumServingTags
                MsgBox "OPC Num Serving Tags = " & sOPCNumServingTags,
                vbInformation, GetProjectTitle
                Set objOPCServe = Nothing
        End If
End Sub
```

GetServerName,OPCServerCmdTarget Function

Syntax GetServerName()

Description This function returns the project's Movicon OPC Server name.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim objOPCServe As OPCServerCmdTarget
        Dim sOPCServerName As String

        Set objOPCServe = GetOPCServer()
        If Not objOPCServe Is Nothing Then
                sOPCServerName = objOPCServe.GetServerName
                MsgBox "OPC Server Name = " & sOPCServerName, vbInformation,
                GetProjectTitle
                Set objOPCServe = Nothing
        End If
End Sub
```

GetNumConnectedClients, OPCServerCmdTarget Function

Syntax GetNumConnectedClients()

Description This function returns the number of OPC clients connect to the project's Movicon OPC Server.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim objOPCServe As OPCServerCmdTarget
        Dim sOPCNumConnectedClients As Long

        Set objOPCServe = GetOPCServer()
        If Not objOPCServe Is Nothing Then
                sOPCNumConnectedClients = objOPCServe.GetNumConnectedClients
```

```
                MsgBox "OPC Num Connected Clients = " & sOPCNumConnectedClients,
                vbInformation, GetProjectTitle
                Set objOPCServe = Nothing
        End If
End Sub
```

ServerStatus, OPCServerCmdTarget Property

Syntax ServerStatus()

Description This property is not managed. The returned value is always "0".

Parameter	Description
None	None

Result Integer

Example:

46. PmeDocCmdTarget

AckAllAlarms, PmeDocCmdTarget Function

Syntax AckAllAlarms

Description Acknowledges (Ack) all the alarms currently present in the Movicon Project.

Parameter	Description
None	None

Result Boolean

Example:
Public Sub Click()
    AckAllAlarms
End Sub

AddSysLogMessage, PmeDocCmdTarget Function

Syntax AddSysLogMessage(_lpszMessage)

Description This method is used for sending system messages to the Movicon log. The messages will then be traced in the 'System' output window and in the Historical Log file.

Parameter	Description
lpszMessage As String	Message text to be printed in Historical Log.

Result Boolean

Example:
Option Explicit
Sub Main
    AddSysLogMessage("Test Message")
End Sub

CreateObjectLic, PmeDocCmdTarget Function

Syntax CreateObjectLic(_lpszServerName, _lpszLicense)

Description Allows you to create objects from basic codes (eg. ActiveX) which need you to get license number. To know the function parameters see "How to use ActiveX requiring a licence".

Parameter	Description
ByVal lpszServerName as String	Name of the server which creates a reference to the ActiveX.
ByVal lpszLicense as String	Runtime or Development licence number for the ActiveX.

Result Object

Example:

```
Sub Main
        Dim Socket As MSWinsockLib.Winsock
        'Set Socket = New MSWinsockLib.Winsock
        Set Socket = CreateObjectLic ("MSWinsock.Winsock.1", "2c49f800-xxx-xxx-xxx-
        0080c7e7b78d")
        If Socket Is Nothing Then Exit Sub
        Socket.Protocol = sckTCPProtocol
        Socket.LocalPort = 1000
        Socket.Listen
End Sub
```

CreateRemoteObject, PmeDocCmdTarget Function

Syntax CreateRemoteObject(_lpszServerName, _lpszLocation)

Description Creates an object linked to a remote server component installed on another computer. Remote Server components are designed to be DCOM standard ready and therefore visible on net.

Parameter	Description
lpszServerName As String	Name of the DCOM server.
lpszLocation As String	Name of the remote computer or IP address.

Result Object

Example:

```
Sub Main
        Dim objRemote As Object
```

```
        Set objRemote =
        CreateRemoteObject("RemoteServerName","RemotePCName")
        …
        …
        Set objRemote = Nothing
End Sub
```

GetAlarm, PmeDocCmdTarget Function

Syntax GetAlarm(_lpszAlarmName)

Description This function returns the alarm object identified by the name in string format (lpszAlarmName parameter). In cases where the alarm has been associated to a variable (alarm as template), you must specify both alarm name and variable name:

GetAlarmObject(<Alarm Name> <Variable Name>)

Parameter	Description
lpszAlarmName As String	Name of the alarm to be retrieved.

Result Object
If Function has been executed successfully it will retrieve an object of type AlarmCmdTarget if otherwise Nothing is returned.

Example:
Example1:

```
        Public Sub Click()
                Dim objAlarm As AlarmCmdTarget
                Dim bResult As Boolean
                Set objAlarm = GetAlarm("Alarm1")
                bResult = objAlarm.Enabled
                Debug.Print bResult
                Set objAlarm = Nothing
        End Sub
```

Example2:

```
        Public Sub Click()
                'Alarm as template
                Dim objAlarm As AlarmCmdTarget
                Dim bResult As Boolean
                Set objAlarm = GetAlarm("Alarm2 Var0001")
                bResult = objAlarm.Enabled
                Debug.Print bResult
                Set objAlarm = Nothing
        End Sub
```

GetAlarmsPath, PmeDocCmdTarget Function

Syntax GetAlarmsPath()

Description Gets a string containing the working folder in which Movicon will file any forthcoming comments associated by the operator to each alarm.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sResult As String
        sResult = GetAlarmsPath
        Debug.Print sResult
End Sub
```

GetChildProject, PmeDocCmdTarget Function

Syntax GetChildProject(_lpszName)

Description Gets the child project object relating to the name specified. This will allow you to access the child project's methods and properties. The returned object is PmeDocCmdTarget type.

Parameter	Description
lpszName As String	Name of object to be returned.

Result Object
If Function has been executed successfully it will retrieve an object of type PmeDocCmdTarget if otherwise Nothing is returned.

Example:

```
Public Sub Click()
        Dim myObject As PmeDocCmdTarget
        Set myObject = GetChildProject("ChildProject1")
        'If String Table has already been inizialised (Se inizializzata la Tabella delle
        Stringhe)
                Debug.Print "Child.ActiveLanguage: " & myObject.ActiveLanguage
        Set mObject = Nothing
End Sub
```

GetCurrentListAlarms, PmeDocCmdTarget Function

Syntax GetCurrentListAlarms()

Description Returns a string containing the currently active alarms. The string will be composed in succession for each alarm by description, by the associated variable name and by the name of the threshold. This function returns an empty string only when all the alarms have been reset.

The character in the string which separates the alarms is "vbLf". Furthermore, the string is composes for each alarm in the following way:

AlarmName|ThresholdName

if the alarm has been managed as template, the string will be:

AlarmName VariableName|ThresholdName

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim aListAlarms() As String
        Dim i As Integer

        aListAlarms() = Split(CStr(GetCurrentListAlarms), vbLf)
        On Error Resume Next
        For i = 0 To UBound(aListAlarms)
                Debug.Print "Alarm" & i & " -> " & aListAlarms(i)
        Next
End Sub
```

GetDataLoggerRecipe, PmeDocCmdTarget Function

Syntax GetDataLoggerRecipe(_lpszName)

Description Gets datalogger object identified by the ipszName parameter.

Parameter	Description
lpszName As String	Name of datalogger to be retrieved.

Result Object
If Function has been executed successfully it will retrieve an object of type DLRCmdTarget if otherwise Nothing is returned.

Example:

```
Public Sub Click()
        Dim mObject As DLRCmdTarget
        Dim bResult As Boolean
        Set mObject = GetDataLoggerRecipe("Log 5 sec")
        bResult = mObject .Enabled
        Debug.Print bResult
        Set mObject = Nothing
End Sub
```

GetDataLoggerRecipePath, PmeDocCmdTarget Function

Syntax GetDataLoggerRecipePath()

Description Gets a string containing the working folder where Movicon will file data recorded by the project's dataloggers. Further information can be found in the chapter on DataLoggers.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sResult As String
        sResult = GetDataLoggerRecipePath
        Debug.Print sResult
End Sub
```

GetDataPath, PmeDocCmdTarget Function

Syntax GetDataPath()

Description Gets a string containing the working folder where Movicon will file data relating to the retentive variables from the Real Time DB.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sResult As String
        sResult = GetDataPath
        Debug.Print sResult
End Sub
```

GetDrawingPath, PmeDocCmdTarget Function

Syntax GetDrawingPath()

Description Gets a string containing the working folder where Movicon will search for bitmap or jpg type images used in the project.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sResult As String
        sResult = GetDrawingPath
        Debug.Print sResult
End Sub
```

GetDriverInterface, PmeDocCmdTarget Function

Syntax GetDriverInterface()

Description Gets you access to the Driver Interface methods and properties.

Parameter	Description
None	None

Result Object
If Function has been executed successfully it will retrieve an object of type DriverInterface if otherwise Nothing is returned.

Example:

```
Public Sub Click()
        Dim myObject As DriverInterface
        Set myObject = GetDriverInterface("PC Adapter")
        Debug.Print myObject.DelayEvents
        Set myObject = Nothing
End Sub
```

GetEvent, PmeDocCmdTarget Function

Syntax GetEvent(_lpszEventName)

Description Gets the Event object identified by the lpszEventName parameter.

Parameter	Description

lpszEventName As String	Name of the event to be retrieved.

Result Object
If Function has been executed successfully it will retrieve an object of type EventCmdTarget if otherwise Nothing is returned.

Example:

```
Public Sub Click()
        Dim obj As EventCmdTarget
        Dim bResult As Boolean
        Set obj = GetEvent("Event")
        bResult = obj.Enabled
        Debug.Print bResult
        Set obj = Nothing
End Sub
```

GetFatherProject, PmeDocCmdTarget Function

Syntax GetFatherProject(_lpszName)

Description Gets the father project object relating to the name specified to allow you to access its methods and properties. The returned object is PmeDocCmdTarget type.

Parameter	Description
lpszName As String	Name of the object to be retrieved.

Result Object
If Function has been executed successfully it will retrieve an object of type PmeDocCmdTarget if otherwise Nothing is returned.

Example:

```
Public Sub Click()
        Dim mObject As PmeDocCmdTarget
        Set mObject = GetFatherProject("ProgettoFiglio1")
        'If String Table has already been inizialised (Se inizializzata la Tabella delle Stringhe)
                Debug.Print "Father.ActiveLanguage: " & mObject.ActiveLanguage
        Set mObject = Nothing
End Sub
```

GetHisLogADOConn, PmeDocCmdTarget Function

Syntax GetHisLogADOConn()

Description This function retrieved the ADODB connection relating to the Project Historical Log Settings.

If used in Windows CE, this function will always return an ADOCE.connection.3.1. type object. Furthermore, avoid

using the "close method" to close ADO connections, otherwise Movicon will no longer be able to access that database.

Parameter	Description
None	None

Result Object
If Function has been executed successfully it will retrieve an object of type ADODB.Connection if otherwise Nothing is returned.

Example:

```
Sub Main
        Dim Conn1 As New ADODB.Connection
        Dim Rs1 As New ADODB.Recordset
        Dim contFields As Integer
        Dim sQuery As String
        Set Conn1 = GetHisLogADOConn
        sQuery = "SELECT * FROM SysMsgs"
        Set Rs1 = CreateObject("ADODB.Recordset")
        Rs1.Open sQuery, Conn1, adOpenForwardOnly, adLockReadOnly,
        ADODB.adCmdText
        ' Loop per stampare tutti i campi del recorset
        While Not Rs1.EOF
                For contFields = 0 To (Rs1.Fields.Count-1)
                        Debug.Print Rs1.Fields(contFields).Name & " = " &
                        Rs1.Fields(contFields).Value
                Next
                Rs1.MoveNext
        Wend
        Rs1.Close
        Conn1.Close
End Sub
```

GetHisLogDNSConnectionString, PmeDocCmdTarget Function

Syntax GetHisLogDNSConnectionString()

Description Returns the name of the ODBC link which Movicon uses for recording all the project or system events occurred during the applied project run. The Historical Log events are recorded on two different database files according to the event type. All the variable tracing events are saved in the "ProjectName_TraceDB.mdb" file in the "DATA" project folder. All the other information is saved in the "ProjectName_HisLog.mdb" Historical Log Database file in the "LOGS" project folder.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sResult As String
        sResult = GetHisLogDNSConnectionString
        Debug.Print sResult
End Sub
```

GetIOPortInterface, PmeDocCmdTarget Function

Syntax GetIOPortInterface

Description Lets you access the IOPortInterface methods and properties for managing communication ports.

Parameter	Description
None	None

Result Object
If Function has been executed successfully it will retrieve an object of type IOPortInterface if otherwise Nothing is returned.

Example:

```
Public Sub Click()
        Dim objIOPort As IOPortInterface
        Dim ID As Long
        Set objIOPort = GetIOPortInterface
        ID = objIOPort.IOPortOpen("COM1:9600,n,8,1")
        Set objIOPort = Nothing
        MsgBox "PortOpen = " & ID, vbInformation, GetProjectTitle
End Sub
```

GetLastAlarmOn, PmeDocCmdTarget Function

Syntax GetLastAlarmOn

Description Allows you to access the methods and properties of the AlarmThresholdCmdTarget relating to the last alarm occurrence.

Parameter	Description
None	None

Result Object
An AlarmThresholdCmdTarget object is returned if the function was executed successfully, otherwise the object will be returned as Nothing.

Example:

```
Option Explicit
Public Sub Click()
```

```
        Dim objAlarmThreshold As AlarmThresholdCmdTarget
        Set objAlarmThreshold = GetLastAlarmOn
        If objAlarmThreshold Is Nothing Then Exit Sub
        MsgBox("LastAlarmThreshold Name is " & objAlarmThreshold.Name,vbOkOnly,
        "")
End Sub
```

GetLogPath, PmeDocCmdTarget Function

Syntax GetLogPath()

Description Gets a string containing the working folder in which Movicon will file data recorded on Historical Log events and project variable tracings. Further information can be found in the chapter on the Historical Log.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sResult As String
        sResult = GetLogPath
        Debug.Print sResult
End Sub
```

GetNetworkClient, PmeDocCmdTarget Function

Syntax GetNetworkClient()

Description Gets you access to the methods and properties described in the NetworkClientCmd section.

Parameter	Description
None	None

Result Object
If Function has been executed successfully it will retrieve an object of type NetworkClientCmd if otherwise Nothing is returned.

Example:

```
Public Sub Click()
        Dim myObject As NetworkClientCmd
        Set myObject = GetNetworkClient()
        myObject.CloseAllConnections
        Set myObject = Nothing
```

```
End Sub
```

GetNetworkRedundancy, PmeDocCmdTarget Function

Syntax GetNetworkRedundancy()

Description Allows you to access the methods and properties described in the NetworkRedudancyCmd. interface.

Parameter	Description
None	None

Result Object
If Function has been executed successfully it will retrieve an object of type NetworkRedudancyCmd if otherwise Nothing is returned.

Example:

```
Public Sub Click()
        Dim objNR As NetworkRedudancyCmd
        Set objNR = GetNetworkRedudancy
        If Not objNR Is Nothing Then
                MsgBox "ActNumRetries = " & objNR.ActNumRetries,vbInformation,GetProjectTitle
        End If
End Sub
```

GetNetworkServer, PmeDocCmdTarget Function

Syntax GetNetworkServer()

Description Gets you access to methods and properties described in the NetworkServerCmd section.

Parameter	Description
None	None

Result Object
If Function has been executed successfully it will retrieve an object of type NetworkClientCmd if otherwise Nothing is returned.

Example:

```
Public Sub Click()
        Dim myObject As NetworkClientCmd
        Set myObject = GetNetworkServer()
        myObject.CloseAllConnections
        Set myObject = Nothing
End Sub
```

GetNetworkUserLogPath, PmeDocCmdTarget Function

Syntax GetNetworkUserLogPath()

Description Returns a string containing the working folder in which Movicon will insert the project's Network users.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sResult As String
        sResult = GetNetworkUserLogPath
        Debug.Print sResult
End Sub
```

GetOPCClient, PmeDocCmdTarget Function

Syntax GetOPCClient()

Description Gets you access to the OPCClientCmdTarget methods and properties.

Parameter	Description
None	None

Result Object
If Function has been executed successfully it will retrieve an object of type OPCClientCmdTarget if otherwise Nothing is returned.

Example:

```
Public Sub Click()
        Dim myObject As OPCClientCmdTarget
        Dim bResult As Long
        Set myObject = GetOPCClient()
        bResult = myObject.TimeoutDynamicOperation
        Debug.Print bResult
        Set myObject = Nothing
End Sub
```

GetOPCServer, PmeDocCmdTarget Function

Syntax GetOPCServer()

Description Gets you access to the OPCServerCmdTarget methods and properties.

This function is not supported in Windows CE.(if used always returns 'null')

Parameter	Description
None	None

Result Object
If Function has been executed successfully it will retrieve an object of type OPCServerCmdTarget if otherwise Nothing is returned.

Example:

```
Public Sub Click()
        Dim myObject As OPCServerCmdTarget
        Dim bResult As String
        Set myObject = GetOPCServer()
        bResult = myObject.ServerStatus
        Debug.Print bResult
        Set myObject = Nothing
End Sub
```

GetProjectFileName, PmeDocCmdTarget Function

Syntax GetProjectFileName()

Description Returns a string with the path and name of the project being run.

Parameter	Description
None	None

Result String

Example:

```
Sub Main
        Dim sResult As String
        sResult = GetProjectFileName
        MsgBox("GetProjectFileName=" & sResult)
End Sub
```

GetProjectTitle, PmeDocCmdTarget Function

Syntax GetProjectTitle()

Description Returns a string with the title assigned to the main folder in the project window.

Parameter	Description
None	None

Result String

Example:

```
Sub Main
        Dim sResult As String
        sResult = GetProjectTitle
        MsgBox("GetProjectTitle=" & sResult)
End Sub
```

GetRealTimeDB, PmeDocCmdTarget Function

Syntax GetRealTimeDB()

Description Allows access to the methods and properties described in the DBVariableCmdTarget, for managing the Movicon RealTimeDB from basic codes.

Parameter	Description
None	None

Result Object
If Function has been executed successfully it will retrieve an object of type DBVariableCmdTarget if otherwise Nothing is returned.

Example:

```
Sub Main
        Dim RealTimeDB As DBVariableCmdTarget
        Set RealTimeDB = GetRealTimeDB
        If RealTimeDB Is Nothing Then Exit Sub
        MsgBox "EnableInUseVarMng is " & RealTimeDB.EnableInUseVarMng
End Sub
```

GetResourcePath, PmeDocCmdTarget Function

Syntax GetResourcePath()

Description Returns a string containing the working folder in which Movicon will save the project's resources (Screens, Menus, Accelerators, etc.,).

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sResult As String
        sResult = GetResourcePath
        Debug.Print sResult
End Sub
```

GetScaling, PmeDocCmdTarget Function

Syntax GetScaling(_lpszScalingName)

Description Returns the Variable Scaling object identified by the lpszScalingName parameter.

Parameter	Description
lpszScalingName As String	Name of variable scaling object to be retrieved.

Result Object
If Function has been executed successfully it will retrieve an object of type ScalingCmdTarget if otherwise Nothing is returned.

Example:

```
Public Sub Click()
        Dim obj As ScalingCmdTarget
        Dim bResult As Boolean
        Set obj = GetScaling("Scaling")
        bResult = obj.Enabled
        Debug.Print bResult
        Set obj = Nothing
End Sub
```

GetScheduler, PmeDocCmdTarget Function

Syntax GetScheduler(_lpszSchedulerName)

Description Returns the Scheduler commands object identified in the lpszScalingName parameter.

Parameter	Description

lpszSchedulerName As String	Name of the scheduler commands object to be retrieved.

Result Object
If Function has been executed successfully it will retrieve an object of type SchedulerCmdTarget if otherwise Nothing is returned.

Example:

```
Public Sub Click()
        Dim obj As SchedulerCmdTarget
        Dim bResult As Boolean
        Set obj = GetScheduler("Scaling")
        bResult = obj.Enabled
        Debug.Print bResult
        Set obj = Nothing
End Sub
```

GetSynopticInterface, PmeDocCmdTarget Function

Syntax GetSynopticInterface()

Description Allows access to the methods and properties described in the SynopticCmdTarget from script resource.

By using the 'GetSynopticInterface' and 'GetSubObject' functions you can create references to screens, and objects within screens.
You must be careful when using these functions within a basic script resource since they are not part of screen context and are used externally. Therefore objects such as the 'SynopticCmdTarget' or 'DrawCmdTarget' created in a screen loaded in memory may get used with these functions when screen is unloaded. If ever this should happen, the basic script will raise an error returning a "ActiveX Automation Error" message interrupting its execution.
To avoid this from happening you should not use these functions in scripts unless absolutely necessary in which case you can handle the error with the basic script's "OnError" functions.

It is strongly advised against using the "GetSynopticInterface" function within design script code. Would be best to use the "GetSynopticObject" function from the "DrawCmdTarget" interface instead.

Caution! This method is not supported on WinCE platforms. This is because it uses API DCOM which may not be supported by the WinCE platform.

Parameter	Description
None	None

Result Object
If Function has been executed successfully it will retrieve an object of type SynopticCmdTargetif otherwise Nothing is returned.

Example:

```
Sub Main
        Dim Screen As SynopticCmdTarget
        Set Screen = GetSynopticInterface("Screen1")
```

```
        If Screen Is Nothing Then Exit Sub
        Screen.BackColor = RGB(34,67,234)
End Sub
```

GetUserAndGroup, PmeDocCmdTarget Function

Syntax GetUserAndGroup()

Description Returns the object relating to the Users and User Groups management. The returned object's methods and properties are those described in the UserAndGroupCmdTarget section.

Parameter	Description
None	None

Result Object
If Function has been executed successfully it will retrieve an object of type UserGroupCmdTarget otherwise Nothing is returned.

Example:

```
Public Sub Click()
        Dim myObject As UserGroupCmdTarget
        Dim bResult As Boolean
        Set myObject = GetUserAndGroup()
        bResult = myObject.EnableAutoLogoff
        Debug.Print bResult
        Set myObject = Nothing
End Sub
```

IsAlarmAreaActive, PmeDocCmdTarget Function

Syntax IsAlarmAreaActive(_lpszServer, _lpszArea)

Description This method allows you to verify whether an alarm area contains active alarms. When the this method returns a "false" value, this means that no alarms are active.

Parameter	Description
lpszServer As String	Name or IP address of Server to be interrogated. When string is empty string the Local Server will be considered.
lpszArea As String	Name of the area to be controlled.

Result Boolean

Example:

```
Sub Main
```

```
        IsAlarmAreaActive("", "Area1")
End Sub
```

IsAlarmAreaON, PmeDocCmdTarget Function

Syntax IsAlarmAreaON(_lpszServer, _lpszArea)

Description This method allows you to verify whether an alarm area contains alarms with the ON status. When this method returns "false" value, this means that no alarm is ON.

Parameter	Description
lpszServer As String	Name or IP address of Server to be interrogated. When the string is empty, the Local Server will be considered.
lpszArea As String	Name of the area to be controlled.

Result Boolean

Example:

```
Sub Main
        IsAlarmAreaON("", "Area1")
End Sub
```

IsChildProject, PmeDocCmdTarget Function

Syntax IsChildProject()

Description Returns the True boolean value when called from a child project script code.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim bResult As Boolean
        bResult = IsChildProject
        Debug.Print "IsChildProject: " & bResult
End Sub
```

IsInStoppingMode, PmeDocCmdTarget Function

Syntax IsInStoppingMode()

Description This function verifies the application's status being the transition from run to the stop mode. Usually used as a loop test.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Main
        ...
        While IsInStoppingMode = 0
        ...
        Wend
        ...
End Sub
```

IsRunning, PmeDocCmdTarget Function

Syntax IsRunning()

Description Verifies project run status.
This function can be invoked from external programmes, comprised of another Movicon project, and gets the project's status: in run mode or not in run mode.
The returned True value indicated that the project in run mode, while the False value indicates that no project is being run.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Main
        Dim mObject As PmeDocCmdTarget
        Set myObject = GetFatherProject("TestPrj")
        'If String Table has already been inizialised (Se inizializzata la Tabella delle Stringhe)
        Debug.Print "Father.Is Running: " & myObject.IsRunning
        Set mObject = Nothing
End Sub
```

PlaySoundFile, PmeDocCmdTarget Function

Syntax PlaySoundFile(_lpszSoundFile, _bAsync, _bLoop)

Description This function is used to play an audio file. Once put into execution the sound reproduction can be interrupted by using the "StopPlaySoundFile" function.

Parameter	Description
lpszSoundFile As String	Path and Name of file to be played.
bAsync As Boolean	Defines whether the function must be played in synchronous or asynchronous mode. When The value is set at False the basic routine will be executed following the complete execution of the audio file only. When set at True, the audio file will be put into execution and the basic routine will immediately continue on to execute the next instructions.
bLoop As Boolean	Defines whether the audio file must be played once only or in a continuous loop. The audio file is played in a continuous loop when the value is set at True only if the "bAsync" parameter is set at True or in other words played in asynchronous mode.

Result Boolean

Example:

```
Option Explicit
Sub Main
        PlaySoundFile("C:\FileName.wav", True, False)
End Sub
```

ResetAllAlarms, PmeDocCmdTarget Function

Syntax ResetAllAlarms

Description Resets all the alarms in the Movicon project that have previously been silenced with ACK.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        ResetAllAlarms
End Sub
```

RunningOnCE, PmeDocCmdTarget Function

Syntax RunningOnCE()

Description Returns the True value when the basic code is being run on Windows CE platform.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Main
        If RunningOnCE Then
        MsgBox("This is a WinCe operating system !")
        End If
End Sub
```

RunScript, PmeDocCmdTarget Function

Syntax RunScript(_lpszScriptName, _dwTimeout, _bSpawnNewInstanceAllowed)

Description Runs the Basic Script function specified as parameter and previously inserted in the Movicon Basic Script resource.
The name of the basic script identified by the lpszScriptName parameter can be run from one or more parameters subdivided by a comma. Therefore the GetParameter function, described in the ScriptMEInterface, is used for retrieving the value of any parameter passed.

Parameter	Description
lpszScriptName As String	Name of script.
dwTimeout As Long	Timeout value for executing the script. This value has meaning only when the script is not set as "Separate Thread".
bSpawnNewInstanceAllowed As Boolean	Enabling of execution of many script instances at the same time. However, it is necessary that the script's "Maximun Instances" property be higher than one.

Result Boolean

Example1:

```
Public Sub Click()
        ...
        RunScript("BasicScript1",100,False)
        ...
End Sub
```

Example2:

```
Public Sub Click()
        ...
```

```
        RunScript("BasicScript1,Param1,Param2",100,False)
        ...
End Sub
```

SendDispatcherMessage, PmeDocCmdTarget Function

Syntax SendDispatcherMessage(_lpszMedia, _lpszMessage, _lpszUser, _nSeverity)

Description This method can be used for sending the dispatcher a message among those supported by the AlarmDispatcher. This method starts the dispatcher in licensed mode if not already started.

Parameter	Description
lpszMedia as String	notification type to send to dispatcher may obtain the following values: “sms”, “email”, “voice”, “smpp”, “gsm”, “smtp”, “mapi” e “fax”.
lpszMessage as String	Message to be sent with the dispatcher. When sending emails the message is enriched with a particular syntax, which included specific data that appear in the email: The email syntax is: from~subject~mail message where "~" is used as a separator char. (ex. geronimo@sioux.augh~Subject test prova~test Message).
lpszUser as String	Name of Movicon project user to be sent message. Movicon retrieves the data necessary from this user to notify the dispatcher to whom the message is to be sent.
nSeverity as Long	Alarm Severty index. This numver is used by the dispatcher to set the message mailing time based on the project's AlarmDispatcher's configuration.

Result Boolean

Example:

```
Option Explicit
Sub Main
        SendDispatcherMessage("sms", "Test message", "User", 1)
End Sub
```

StartAlarmDispatcher, PmeDocCmdTarget Function

Syntax StartAlarmDispatcher

Description This method is used for executing the AlarmDispatcher by automatically enabling its license. Obviously the 'AlarmDispatcher is only executed when there is a corresponding license option otherwise this message will show:

"Your license restriction doesn't allow to run the Alarm Dispatcher"

This method is useful for those who wish to use the AlarmDispatcher in

custom mode with VBA codes without using the Movicon alarm functions.

TIP: Movicon makes a reference to the object which the 'AlarmDispatcher creates in the ROT. Therefore this method can be used only once, after the project is executed, to start the dispatcher.

Parameter	Description
None	None

Result Boolean

Example:
Option Explicit
Sub Main
StartAlarmDispatcher
End Sub

StopPlaySoundFile, PmeDocCmdTarget Function

Syntax StopPlaySoundFile()

Description This function is used for stopping an audio file from playing which was originally executed with the "PlaySoundFile" function.

Parameter	Description
None	None

Result Boolean

Example:
Option Explicit
Sub Main
PlaySoundFile("C:\FileName.wav", True, True)
...
StopPlaySoundFile
End Sub

UnloadScript, PmeDocCmdTarget Function

Syntax UnloadScript(_lpszScriptName)

Description Unloads the script identified by the lpszScriptName parameter.

Parameter	Description

lpszScriptName As String	Name of script.

Result Boolean

Example:

```
Public Sub Click()
        ...
        UnloadScript("Basic Script1")
        ...
End Sub
```

ActiveLanguage, PmeDocCmdTarget Property

Syntax ActiveLanguage = _String

Description Permits the active language to be set or read.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sLanguage As String
        sLanguage = ActiveLanguage
        Debug.Print sLanguage
End Sub
```

ChildProjectActiveNetworkServer, PmeDocCmdTarget Property

Syntax ChildProjectActiveNetworkServer = _String

Description this property returns the network Server name or IP address from which the child project will retrieve data. This property is read only.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
```

```
        MsgBox "Active Child Network Server = " &
        GetChildProject("ChildProject1").ChildProjectActiveNetworkServer,
        vbInformation, GetProjectTitle
End Sub
```

ChildProjectBackupNetworkServer, PmeDocCmdTarget Property

Syntax ChildProjectBackupNetworkServer = _String

Description This property returns the Backup Network Server name or IP address from which the Child project will retrieve data. This property is read only.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        MsgBox "Child Network Backup Server = " &
        GetChildProject("ChildProject1").ChildProjectBackupNetworkServer,
        vbInformation, GetProjectTitle
End Sub
```

ChildProjectName, PmeDocCmdTarget Property

Syntax ChildProjectName = _String

Description This property returns the name of the child project. In cases where no value has been inserted in the "Child Project Name" property from the "Child Project Options" group, this property will return an empty string. This property is read only.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        MsgBox "Child Project Name = " & GetChildProject("ChildProject1").ChildProjectName,
        vbInformation, GetProjectTitle
End Sub
```

ChildProjectNetworkServer, PmeDocCmdTarget Property

Syntax ChildProjectNetworkServer = _String

Description This property returns the name of the Network Server project from which the child project retrieves data. This property is read only.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        MsgBox "Child Project Network Server = " &
        GetChildProject("ChildProject1").ChildProjectNetworkServer, vbInformation,
        GetProjectTitle
End Sub
```

ChildProjectStartable, PmeDocCmdTarget Property

Syntax ChildProjectStartable = _Boolean

Description This property allows you to check whether the child project is "Startable" or not. When a child project is startable this means all the Server side of the child project will be started up, being the Driver, OPC, Networking, etc. When a project is not startable you can always get access to the child project screens through the Parent project. This property is read only.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        MsgBox "Child Project Startable = " &
        GetChildProject("ChildProject1").ChildProjectStartable, vbInformation,
        GetProjectTitle
End Sub
```

HisLogAlarmDurationDays, PmeDocCmdTarget Property

Syntax HisLogAlarmDurationDays = _Long

Description Allows you to set or read how long the alarm messages are to be kept in the database for. The Default setting is 180 days which can be changed as required by also taking into account the number of recordings to be done in the set time interval.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim sResult As Long
        sResult = HisLogAlarmDurationDays
        Debug.Print sResult
End Sub
```

HisLogAlarmDurationHours, PmeDocCmdTarget Property

Syntax HisLogAlarmDurationHours = _Long

Description Allows you to set or read how long, in hours, the alarm messages are to be kept in the database for.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim sResult As Long
        sResult = HisLogAlarmDurationHours
        Debug.Print sResult
End Sub
```

HisLogAlarmDurationMinutes, PmeDocCmdTarget Property

Syntax HisLogAlarmDurationMinutes = _Long

Description Allows you to set or read how long in minutes the alarm messages are to be kept in the database for.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim sResult As Long
        sResult = HisLogAlarmDurationMinutes
        Debug.Print sResult
End Sub
```

HisLogAlarmTable, PmeDocCmdTarget Property

Syntax HisLogAlarmTable = _String

Description Sets or returns the name of the Historical Log table which will contain the messages inherent to the project's alarms.

This property's modification will be acquired only if done during project development mode, for example using Symbol Dropping Code. In this case, the property will be modified in the project statically. Once the project has been started up in Runtime, any modification to this property will be ignored.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sResult As String
        sResult = HisLogAlarmTable
        Debug.Print sResult
End Sub
```

HisLogCommentColName, PmeDocCmdTarget Property

Syntax HisLogCommentColName

Description This property sets or returns the name of the Historical Log tables' Comment Column.

This property's modification will be acquired only if done during project development mode, for example using a Symbol's Dropping Code. In this case, the property will be modified in the project statically. Once the project has been started up in Runtime, modification to this property will be ignored.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sResult As String
        sResult = HisLogCommentColName
        Debug.Print sResult
End Sub
```

HisLogDefVarCharPrecision, PmeDocCmdTarget Property

Syntax HisLogDefVarCharPrecision = _Long

Description Sets or returns the maximum precision for the string type columns. The number set represents the number of string characters.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim sResult As Long
        sResult = HisLogDefVarCharPrecision
        Debug.Print sResult
End Sub
```

HisLogDescriptionColName, PmeDocCmdTarget Property

Syntax HisLogDescriptionColName

Description This property sets or returns the name of the Historical Log tables' Description Column. Default name will be used if nothing has been specified. The Description Column shows the record event description.

This property's modification will be acquired only if done during project development mode, for example using a Symbol's Dropping Code. In this case, the property will be modified in the project statically. Once the project has been started up in Runtime, modification to this property will be ignored.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sResult As String
        sResult = HisLogDescriptionColName
        Debug.Print sResult
End Sub
```

HisLogDriverDurationDays, PmeDocCmdTarget Property

Syntax HisLogDriverDurationDays = _Long

Description Allows you to set or read how long the Communication Driver messages are to be kept in the database for. The default setting is 180 days which can be changed as required taking into account the number of recordings to be done in the set interval time.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim sResult As Long
        sResult = HisLogDriverDurationDays
        Debug.Print sResult
End Sub
```

HisLogDriverDurationHours, PmeDocCmdTarget Property

Syntax HisLogDriverDurationHours = _Long

Description Allows you to set or read how long, in hours, the Communication Driver messages are to be kept in the Database for.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim sResult As Long
        sResult = HisLogDriverDurationHours
        Debug.Print sResult
End Sub
```

HisLogDriverDurationMinutes, PmeDocCmdTarget Property

Syntax HisLogDriverDurationMinutes = _Long

Description Allows you to set or read how long, in minutes, the Communication Driver messages are to be kept in the Database for.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim sResult As Long
        sResult = HisLogDriverDurationMinutes
        Debug.Print sResult
End Sub
```

HisLogDriverTable, PmeDocCmdTarget Property

Syntax HisLogDriverTable = _String

Description Sets or returns the name of the Historical Log table containing the project's Drivers' inherent messages.

This property's modification will be acquired only if done during project development mode, for example using a Symbol's Dropping Code. In this case, the property will be modified in the project statically. Once the project has been started up in Runtime, modifications to this property will be ignored.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sResult As String
        sResult = HisLogDriverTable
        Debug.Print sResult
End Sub
```

HisLogDsn, PmeDocCmdTarget Property

Syntax HisLogDsn = _String

Description This setting permits you to set or retrieve a customized ODBC link for the Historical Log. Movicon will create a DSN for default with the same project name and "_HisLog" suffix, configured for accessing the database using the project's specified "Default ODBC PlugIN". The DSN name will be:

ProjectName_HisLog

This property is used for customizing the **ODBC** link, creating a customized database that is also different from the one defined in the project's "Default ODBC plugIn".

This property can be used in write only in project design mode, for example dropping Template symbol code. In Runtime mode, however even though it can be modified it will not be applied to the Historical Log which will continue recording using the initial DSN.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sResult As String
        sResult = HisLogDsn
        Debug.Print sResult
End Sub
```

HisLogDurationColName, PmeDocCmdTarget Property

Syntax HisLogDurationColName

Description This property sets or returns the name of the Historical Log tables' Duration Column. Default name will be specified if left blank. The Duration Column shows the duration time of the event in question.

This property's modification will be acquired only if done during project development mode, for example using a Symbol's Dropping Code. In this case, the property will be modified in the project statically. Once the project has been started up in Runtime, modifications to this property will be ignored..

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sResult As String
        sResult = HisLogDurationColName
        Debug.Print sResult
End Sub
```

HisLogEventTypeColName, PmeDocCmdTarget Property

Syntax HisLogEventTypeColName

Description This property sets or returns the name of the Historical Log tables' Event Column. the default name will be used if left unspecified. The Event Column indicates the record event type (i.e. Alarm ON, Alarm OFF, System, etc.)

This property's modification will be acquired only if done during project development mode, for example using a Symbol's Dropping Code. In this case, the property will

be modified in the project statically. Once the project has been started up in Runtime, modifications to this property will be ignored.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sResult As String
        sResult = HisLogEventTypeColName
        Debug.Print sResult
End Sub
```

HisLogEventTypeNumColName, PmeDocCmdTarget Property

Syntax HisLogEventTypeNumColName

Description This property sets or returns the name of the Historical Log tables' Event Type Number Column. If not specified the default name will be used instead. The Event Type Number Column reports the number indicating the record event type.

This property's modification will be acquired only if done during project development mode, for example using a Symbol's Dropping Code. In this case, the property will be modified in the project statically. Once the project has been started up in Runtime, modifications to this property will be ignored.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sResult As String
        sResult = HisLogEventTypeNumColName
        Debug.Print sResult
End Sub
```

HisLogLocalTimeColName, PmeDocCmdTarget Property

Syntax HisLogLocalTimeColName

Description This property sets or returns the name of the Historical Log table Local Time Column. If not specified, the default name will be used instead. The Local Time Column indicates the date and time recording took place referring to local time.

This property's modification will be acquired only if done during project development mode, for example using a Symbol's Dropping Code. In this case, the property will be modified in the project statically. Once the project has been started up in Runtime, modifications to this property will be ignored.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sResult As String
        sResult = HisLogLocalTimeColName
        Debug.Print sResult
End Sub
```

HisLogMaxCacheBeforeFlush, PmeDocCmdTarget Property

Syntax HisLogMaxCacheBeforeFlush = _Long

Description Sets or returns the maximum size of the Cache before the system unloads data on file. The number set is expressed in Bytes.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim sResult As Long
        sResult = HisLogMaxCacheBeforeFlush
        Debug.Print sResult
End Sub
```

HisLogMaxError, PmeDocCmdTarget Property

Syntax HisLogMaxError = _Long

Description Sets or returns the maximum number of DBMS errors that when exceeded the connection will be no longer valid and disconnected and data will be saved on file in ASCII format in the relevant project folders (("DLOGGERS", "LOGS", "DATA").

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim sResult As Long
        sResult = HisLogMaxError
        Debug.Print sResult
End Sub
```

HisLogMaxNumberTrans, PmeDocCmdTarget Property

Syntax HisLogMaxNumberTrans = _Long

Description Sets or returns the maximum number of transitions per cycle to be updated before being closed.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim sResult As Long
        sResult = HisLogMaxNumberTrans
        Debug.Print sResult
End Sub
```

HisLogMSecColName, PmeDocCmdTarget Property

Syntax HisLogMSecColName

Description This property sets or returns the name of the Historical Log tables' MSec Column. If not specified, the default name will be used instead. The MSec Column indicates the milliseconds relating to time of recording.

Modification to this property will be acquired only if done during project development mode, for example using a Symbol's Dropping Code. In this case, the property will be modified in the project statically. Once the project has been started up in Runtime, modifications to this property will be ignored.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sResult As String
        sResult = HisLogMSecColName
        Debug.Print sResult
End Sub
```

HisLogRecycleDBConnection, PmeDocCmdTarget Property

Syntax HisLogRecycleDBConnection = _Boolean

Description Sets or returns the value of the "Keep the DB Connection open" property. When set at True, the DBMS connection will be kept open and used for all the transitions to be executed. When disabled, the DBMS connection will open when a transition is requested and then closed again.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim sResult As Boolean
        sResult = HisLogRecycleDBConnection
        Debug.Print sResult
End Sub
```

HisLogSubEventTypeColName, PmeDocCmdTarget Property

Syntax HisLogSubEventTypeColName

Description This property sets or returns the name of the Historical Log tables' Sub Event Type column. If not specified, the default name will be used instead. The Sub Event Type Column shows the value associated to the "_Sys_HisLogSubEventType_" system variable if in the Real Time DB.

Modification to this property will be acquired only if done during project development mode, for example using a Symbol's Dropping Code. In this case, the property will be modified in the project statically. Once the project has been started up in Runtime, modifications to this property will be ignored.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sResult As String
        sResult = HisLogSubEventTypeColName
        Debug.Print sResult
End Sub
```

HisLogSysTable, PmeDocCmdTarget Property

Syntax HisLogSysTable = _String

Description Sets or returns the name of the Historical Log table containing the project's 'System Messages' messages.

Modification to this property will be acquired only if done during project development mode, for example using a Symbol's Dropping Code. In this case, the property will be modified in the project statically. Once the project has been started up in Runtime, modifications to this property will be ignored.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sResult As String
        sResult = HisLogSysTable
        Debug.Print sResult
End Sub
```

HisLogSystemsDurationDays, PmeDocCmdTarget Property

Syntax HisLogSystemDurationDays = _Long

Description Allows you to set or read how long the System messages are to be kept in the database for. The Default setting is 180 days, but can be changed as required by also taking into account the number of recordings executed in the specified time interval.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim sResult As Long
        sResult = HisLogSystemDurationDays
        Debug.Print sResult
End Sub
```

HisLogSystemsDurationHours, PmeDocCmdTarget Property

Syntax HisLogSystemDurationHours = _Long

Description Allows you to set or get the time duration in hours that the System messages are to kept in the database.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim sResult As Long
        sResult = HisLogSystemDurationHours
        Debug.Print sResult
```

```
End Sub
```

HisLogSystemsDurationMinutes, PmeDocCmdTarget Property

Syntax HisLogSystemDurationMinutes = _Long

Description Allows you to set or get the time duration in minutes that the System messages are to be kept in the database.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim sResult As Long
        sResult = HisLogSystemDurationMinutes
        Debug.Print sResult
End Sub
```

HisLogTimeColName, PmeDocCmdTarget Property

Syntax HisLogTimeColName

Description This property sets or returns the name of the Historical Log tables' Local time Column. If not specified the default name will be used instead. The Local Time Column indicates the local date and time the recording took place.

Modification to this property will be acquired only if done during project development mode, for example using a Symbol's Dropping Code. In this case, the property will be modified in the project statically. Once the project has been started up in Runtime, modifications to this property will be ignored.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sResult As String
```

```
        sResult = HisLogTimeColName
        Debug.Print sResult
End Sub
```

HisLogTransactionIDColName, PmeDocCmdTarget Property

Syntax HisLogTransactionIDColName

Description This property sets or returns the name of the Historical Log tables' Transaction ID Column.

Modification to this property will be acquired only if done during project development mode, for example using a Symbol's Dropping Code. In this case, the property will be modified in the project statically. Once the project has been started up in Runtime, modifications to this property will be ignored.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sResult As String
        sResult = HisLogTransactionIDColName
        MsgBox "HisLogTransactionIDColName = " & sResult, vbInformation, GetProjectTitle
End Sub
```

HisLogUniqueIDColName, PmeDocCmdTarget Property

Syntax HisLogUniqueIDColName

Description This property sets or returns the name of the Historical Log tables' Unique ID Column.

Modification to this property will be acquired only if done during project development mode, for example using a Symbol's Dropping Code. In this case, the property will be modified in the project statically. Once the project has been started up in Runtime, modifications to this property will be ignored.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sResult As String
        sResult = HisLogUniqueIDColName
        MsgBox "HisLogTransactionIDColName = " & sResult, vbInformation,
        GetProjectTitle
End Sub
```

HisLogUseIMDB, PmeDocCmdTarget Property

Syntax HisLogUseIMDB = _Boolean

Description This property is read only and allows you to know if the project's historical log has been set to record values with the InMemoryDataBase engine.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Main
MsgBox "HisLogUseIMDB->" & HisLogUseIMDB
End Sub
```

HisLogUser, PmeDocCmdTarget Property

Syntax HisLogUser = _String

Description Sets or returns the user name used for the ODBC connection.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sResult As String
        sResult = HisLogUser
        Debug.Print sResult
```

```
End Sub
```

HisLogUserColName, PmeDocCmdTarget Property

Syntax HisLogUserColName

Description This property sets or returns the name of the Historical Log table User Column. If not specified, the default name will be used insteas. The User Column indicates the name of the active user when recording took place.

Modification to this property will be acquired only if done during project development mode, for example using a Symbol's Dropping Code. In this case, the property will be modified in the project statically. Once the project has been started up in Runtime, modifications to this property will be ignored.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sResult As String
        sResult = HisLogUserColName
        Debug.Print sResult
End Sub
```

ShutdownScript, PmeDocCmdTarget Property

Syntax ShutdownScript = _String

Description Sets or returns the name of the script to be run when project shuts down.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sResult As String
        sResult = ShutdownScript
        Debug.Print sResult
```

```
End Sub
```

StartChildProjectWithFather, PmeDocCmdTarget Property

Syntax StartChildProjectWithFather = _Boolean

Description This property consents you to check whether the child project was startup together with its parent project or not. When enabled, this property permits the child project to automatically startup when its parent project is put into run mode. This setting only works when the "Startable" has been selected. This property is read only.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        MsgBox "Child Project Start with Father = " &
        GetChildProject("ChildProject1").StartChildProjectWithFather, vbInformation,
        GetProjectTitle
End Sub
```

StartupScreen, PmeDocCmdTarget Property

Syntax StartupScreen = _String

Description Sets or returns the name of the Startup screen.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sResult As String
        sResult = StartupScreen
        Debug.Print sResult
End Sub
```

StartupScript, PmeDocCmdTarget Property

Syntax StartupScript = _String

Description Sets or returns the name of the Startup script.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sResult As String
        sResult = StartupScript
        Debug.Print sResult
End Sub
```

StoreCryptProject, PmeDocCmdTarget Property

Syntax StoreCryptProject = _Boolean

Description Sets or returns the value of the 'Crypt Project' property. When set at true, the project will be saved in crypt format, making it impossible to open the project with any other editor that is not Movicon's.

This property's settings have no effect in runtime. However, if this property is used in a basic script launched from the design mode, it will obtain the new set value in the project.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim sResult As Boolean
        sResult = StoreCryptProject
        Debug.Print sResult
End Sub
```

StoreCryptProjectResources, PmeDocCmdTarget Property

Syntax StoreCryptProjectResources = _Boolean

Description Sets or returns the value of the 'Crypt Project Resources' property. When set to true, the project's resource files will be saved in crypt format, making it impossible to open them with any other editor that is not Movicon's.

This property's settings have no effect in runtime. However, if this property is used in a basic script launched from the design mode, it will obtain the new set value in the project.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim sResult As Boolean
        sResult = StoreCryptProjectResources
        Debug.Print sResult
End Sub
```

StoreCryptProjectStrings, PmeDocCmdTarget Property

Syntax StoreCryptProjectStrings = _Boolean

Description Set or returns the "Tabella Stringhe Criptata" property value. When set to true, the project's String Table files will be saved in cripted format and therefore ONLY the Movicon editor can be used for opening them.

This property's settings have no effect in runtime. However, if this property is used in a basic script launched from the design mode, it will obtain the new set value in the project.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Main
        Dim sResult As Boolean
        sResult = StoreCryptProjectStrings
        Debug.Print sResult
End Sub
```

StoreUnicodeProject, PmeDocCmdTarget Property

Syntax StoreUnicodeProject = _Boolean

Description Sets or returns the 'Unicode Project' property value. When set at true, the entire project will be saved in UTF-16 unicode.

This property's settings have no effect in runtime. However, if this property is used in a basic script launched from the design mode, it will obtain the new set value in the project.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim sResult As Boolean
        sResult = StoreUnicodeProject
        Debug.Print sResult
End Sub
```

StoreZippedProject, PmeDocCmdTarget Property

Syntax StoreZippedProject = _Boolean

Description Sets or returns the 'Zipped Project' property value. When set at true, the entire project will be saved in zip format. The zipping and unzipping of files will automatically be managed by Movicon in real-time, and therefore transparent to the programmer.

This property's settings have no effect in runtime. However, if this property is used in a basic script launched from the design mode, it will obtain the new set value in the project.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim sResult As Boolean
```

```
        sResult = StoreZippedProject
        Debug.Print sResult
End Sub
```

StringFromID, PmeDocCmdTarget Property

Syntax StringFromID(_lpszID) = _String

Description Returns the string, in function with the active column (Language), from string resource, by entering its ID.

Parameter	Description
lpszID As String	ID of the requested string.

Result String

Example:

```
Sub Main
        Dim test As string
        test = StringFromID ("STR0001")
        MsgBox test, 64,"String"
        StringFromID ("STR0001") = "Modificata"
        test = StringFromID ("STR0001")
        MsgBox test, 64,"String"
End Sub
```

TargetClientJ2ME, PmeDocCmdTarget Property

Syntax TargetClientJ2ME = _Boolean

Description Sets or returns the 'Client J2ME' property value. The project development is enabled for J2ME Client platforms (JavaPhones) when this property is set at True.
TIP: multiple platform selecting always involves the functions of the more powerful platform.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim sResult As Boolean
        sResult = TargetClientJ2ME
        Debug.Print sResult
End Sub
```

TargetClientJ2SE, PmeDocCmdTarget Property

Syntax TargetClientJ2SE = _Boolean

Description Sets or returns the value of the 'Client J2SE' property. The project development is enabled for J2SE client platforms (Linux) when this property is set at true.
TIP: multiple platform selecting always involves the functions of the more powerful platform.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim sResult As Boolean
        sResult = TargetClientJ2SE
        Debug.Print sResult
End Sub
```

TargetClientWin32, PmeDocCmdTarget Property

Syntax TargetClientWin32 = _Boolean

Description Sets or returns the value of the "Client Windows XP/Vista/7" property. The project development is enabled for Windows XP/Vista/7 client platforms when this property is set at true.
TIP: multiple platform selecting always involves the functions of the more powerful platform.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim sResult As Boolean
        sResult = TargetClientWin32
        Debug.Print sResult
End Sub
```

TargetClientWinCE, PmeDocCmdTarget Property

Syntax TargetClientWinCE = _Boolean

Description Sets or returns the value of the 'Client WinCE' property. Project development is enabled for Windows CE client platforms when this property is set at true.
TIP: multiple platform selecting always involves the functions of the more powerful platform.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim sResult As Boolean
        sResult = TargetClientWinCE
        Debug.Print sResult
End Sub
```

TargetPlatformWin32, PmeDocCmdTarget Property

Syntax TargetPlatformWin32 = _Boolean

Description Sets or returns the value of the "Windows 32/64 bit" property. Project development is enabled for Windows 32/64 bit when this property is set at True.
TIP: multiple platform selecting always involves the functions of the more powerful platform.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim sResult As Boolean
        sResult = TargetPlatformWin32
        Debug.Print sResult
End Sub
```

TargetPlatformWinCE, PmeDocCmdTarget Property

Syntax TargetPlatformWinCE = _Boolean

Description Sets or returns the value of the 'Platform WinCE' property. Project development is enabled for Windows CE when this property is set at True. TIP: multiple platform selecting always involves the functions of the more powerful platform.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim sResult As Boolean
        sResult = TargetPlatformWinCE
        Debug.Print sResult
End Sub
```

47. RASStationInterface

GetXMLSettings, RASStationInterface Function

Syntax GetXMLSettings()

Description This function returns the settings string in XML format, of the RAS Station inserted in the project.

Warning! This property is also available in the Communication Drivers' Basic Script interface. Even though these two properties have the same function they will not be confused with each other as they are used in two different contexts.

Parameter	Description
None	None

Result String

Example:

```
   Dim RasObj As RASStationInterface
   Dim NetwObj As NetworkClientCmd
Public Sub Click()
        If NetwObj Is Nothing Then Set NetwObj = GetNetworkClient
        If RasObj Is Nothing And Not NetwObj Is Nothing Then Set RasObj =
        NetwObj.GetRASStation("FirstRAS")
        If Not RasObj Is Nothing Then
            MsgBox RasObj.GetXMLSettings,vbOkOnly,GetProjectTitle
        End If
End Sub
```

ConnectionVariable, RASStationInterface Property

Syntax ConnectionVariable = _String

Description This property allows you to read or set the connection variable used by a RAS station to manage calls on command.

Parameter	Description
None	None

Result String

Example:

```
Sub Main
        Dim objRAS As RASStationInterface
        Dim sNewVariable As String

        Set objRAS = GetNetworkClient.GetRASStation("RAS Station00001")
        GetVariableNameFromList(sNewVariable)
        Debug.Print "Before->" & objRAS.ConnectionVariable
        objRAS.ConnectionVariable = sNewVariable
        Debug.Print "Next->" & objRAS.ConnectionVariable
End Sub
```

DisconnectAfterSecs, RASStationInterface Property

Syntax DisconnectAfterSecs = _Long

Description This property is used for setting the communication inactivity time after which the connection will automatically close. The time count starts the moment all the variable connected to the Server are no longer in use.

Warning! This property is also available in the Communication Drivers' Basic Script interface. Even though these two properties have the same function they will not be confused with each other as they are used in two different contexts.

Parameter	Description
None	None

Result Long

Example:

```
   Dim RasObj As RASStationInterface
   Dim NetwObj As NetworkClientCmd
Public Sub Click()
        If NetwObj Is Nothing Then Set NetwObj  = GetNetworkClient
        If RasObj  Is Nothing And Not NetwObj Is Nothing Then Set RasObj  =
        NetwObj.GetRASStation("FirstRAS")
        If Not RasObj Is Nothing Then
                RasObj.DisconnectAfterSecs = 10
        End If
End Sub
```

EndConnectionTime, RASStationInterface Property

Syntax EndConnectionTime = _Date

Description This property returns the time and day in which the connection ended.

Warning! This property is also available in the Communication Drivers' Basic Script interface. Even though these two properties have the same function they will not be confused with each other as they are used in two different contexts.

Parameter	Description
None	None

Result Date

Example:

```
   Dim RasObj As RASStationInterface
   Dim NetwObj As NetworkClientCmd
Public Sub Click()
        If NetwObj Is Nothing Then Set NetwObj = GetNetworkClient
        If RasObj Is Nothing And Not NetwObj Is Nothing Then Set RasObj =
        NetwObj.GetRASStation("FirstRAS")
        If Not RasObj Is Nothing Then
            Debug.Print RasObj.EndConnectionTime
        End If
End Sub
```

IsConnected, RASStationInterface Property

Syntax IsConnected = _Boolean

Description This property returns the status of the connected referenced RAS station. The True boolean value will be returned if connected.

Parameter	Description
None	None

Result Boolean

Example:

```
   Dim RasObj As RASStationInterface
   Dim NetwObj As NetworkClientCmd
Public Sub Click()
        If NetwObj Is Nothing Then Set NetwObj = GetNetworkClient
        If RasObj Is Nothing And Not NetwObj Is Nothing Then Set RasObj =
        NetwObj.GetRASStation("FirstRAS")
        If Not RasObj Is Nothing Then
            RasObj.DisconnectAfterSecs = 10
        End If
End Sub
```

LastConnectionTime, RASStationInterface Property

Syntax LastConnectionTime = _Date

Description This property returns the time and date of the last connection.

Warning! This property is also available in the Communication Drivers' Basic Script interface. Even though both these properties have the same function they will not get mixed up with each other as they are used in two different contexts.

Parameter	Description
None	None

Result Date

Example:

```
   Dim RasObj As RASStationInterface
   Dim NetwObj As NetworkClientCmd
Public Sub Click()
        If NetwObj Is Nothing Then Set NetwObj  = GetNetworkClient
        If RasObj  Is Nothing And Not NetwObj Is Nothing Then Set RasObj  =
        NetwObj.GetRASStation("FirstRAS")
        If Not RasObj Is Nothing Then
                Debug.Print RasObj.LastConnectionTime
        End If
End Sub
```

LastRASErrorNumber, RASStationInterface Property

Syntax LastRASErrorNumber = _Long

Description This property returns the number of the last verified error for the referenced RAS connection.

Warning! This property is also available in the Communication Drivers' Basic Script interface. Even though both these properties have the same function they will not get mixed up with each other as they are used in two different contexts.

Parameter	Description
None	None

Result Long

Example:

```
   Dim RasObj As RASStationInterface
```

```
   Dim NetwObj As NetworkClientCmd
Public Sub Click()
        If NetwObj Is Nothing Then Set NetwObj  = GetNetworkClient
        If RasObj  Is Nothing And Not NetwObj Is Nothing Then Set RasObj  =
        NetwObj.GetRASStation("FirstRAS")
        If Not RasObj Is Nothing Then
            Debug.Print RasObj.LastRASErrorNumber
        End If
End Sub
```

LastRASErrorString, RASStationInterface Property

Syntax LastRASErrorString = _String

Description This property returns the string of the last verified error for the referenced RAS connection.

Warning! This property is also available in the Communication Drivers' Basic Script interface. Even though both these properties have the same function they will not get mixed up with each other as they are used in two different contexts.

Parameter	Description
None	None

Result String

Example:

```
   Dim RasObj As RASStationInterface
   Dim NetwObj As NetworkClientCmd
Public Sub Click()
        If NetwObj Is Nothing Then Set NetwObj  = GetNetworkClient
        If RasObj  Is Nothing And Not NetwObj Is Nothing Then Set RasObj  =
        NetwObj.GetRASStation("FirstRAS")
        If Not RasObj Is Nothing Then
            Debug.Print RasObj.LastRASErrorString
        End If
End Sub
```

NumRetries, RASStationInterface Property

Syntax NumRetries = _Byte

Description This property allows you to set the maximum number of retries to be executed when the first call fails.

Warning! This property is also available in the Communication Drivers' Basic Script interface. Even though both these properties have the same function they will not get mixed up with each other as they are used in two different contexts.

Parameter	Description
None	None

Result Byte

Example:

```
   Dim RasObj As RASStationInterface
   Dim NetwObj As NetworkClientCmd
Public Sub Click()
        If NetwObj Is Nothing Then Set NetwObj = GetNetworkClient
        If RasObj Is Nothing And Not NetwObj Is Nothing Then Set RasObj =
        NetwObj.GetRASStation("FirstRAS")
        If Not RasObj Is Nothing Then
            Debug.Print RasObj.NumRetries
        End If
End Sub
```

Password, RASStationInterface Property

Syntax Password = _String

Description This property allows you to set the user password with which the Client station must be validated by the Server station. This field can be left blank if the "Connection" property has been filled in.

Warning! This property is also available in the Communication Drivers' Basic Script interface. Even though both these properties have the same function they will not get mixed up with each other as they are used in two different contexts.

Parameter	Description
None	None

Result String

Example:

```
   Dim RasObj As RASStationInterface
   Dim NetwObj As NetworkClientCmd
Public Sub Click()
        If NetwObj Is Nothing Then Set NetwObj = GetNetworkClient
        If RasObj Is Nothing And Not NetwObj Is Nothing Then Set RasObj =
        NetwObj.GetRASStation("FirstRAS")
        If Not RasObj Is Nothing Then
            Debug.Print RasObj.Password
        End If
End Sub
```

PhoneBookEntry, RASStationInterface Property

Syntax PhoneBookEntry = _String

Description This property allows you to insert the name of any eventual RAS Connectionto be used for connecting the Client to the Server. In this case the RAS Connection must be one of those configured in the Operating System. When this field is left black you will need to fill in the "Telephone Number", User Name" and "Password" fields.

Warning! This property is also available in the Communication Drivers' Basic Script interface. Even though both these properties have the same function they will not get mixed up with each other as they are used in two different contexts.

Parameter	Description
None	None

Result String

Example:

```
   Dim RasObj As RASStationInterface
   Dim NetwObj As NetworkClientCmd
Public Sub Click()
        If NetwObj Is Nothing Then Set NetwObj = GetNetworkClient
        If RasObj Is Nothing And Not NetwObj Is Nothing Then Set RasObj =
        NetwObj.GetRASStation("FirstRAS")
        If Not RasObj Is Nothing Then
            Debug.Print RasObj.PhoneBookEntry
        End If
End Sub
```

PhoneNumber, RASStationInterface Property

Syntax PhoneNumber= _String

Description This property allows you to set the telephone number which the Client Station must dial. The number must correspond to the line connected to the Server station.
If the "Connection" property have been filled in, this field can be left empty.

Warning! This property is also available in the Communication Drivers' Basic Script interface. Even though both properties have the same function, they will not get mixed up with each other as they are used in two different contexts.

Parameter	Description
None	None

Result String

Example:

```
   Dim RasObj As RASStationInterface
   Dim NetwObj As NetworkClientCmd
Public Sub Click()
        If NetwObj Is Nothing Then Set NetwObj = GetNetworkClient
        If RasObj Is Nothing And Not NetwObj Is Nothing Then Set RasObj =
        NetwObj.GetRASStation("FirstRAS")
        If Not RasObj Is Nothing Then
            Debug.Print RasObj.PhoneNumber
        End If
End Sub
```

PromptForConnection, RASStationInterface Property

Syntax PromptForConnection = _Boolean

Description When this property is enabled a confirmation window will display at the beginning of each connection where the operator must confirm or cancel the call.

Warning! This property is also available in the Communication Drivers' Basic Script interface. Even though both properties have the same function, they will not get mixed up with each other as they to are used in two different contexts.

Parameter	Description
None	None

Result Boolean

Example:

```
   Dim RasObj As RASStationInterface
   Dim NetwObj As NetworkClientCmd
Public Sub Click()
        If NetwObj Is Nothing Then Set NetwObj = GetNetworkClient
        If RasObj Is Nothing And Not NetwObj Is Nothing Then Set RasObj =
        NetwObj.GetRASStation("FirstRAS")
        If Not RasObj Is Nothing Then
            RasObj.PromptForConnection = True
        End If
End Sub
```

RetryAfterSecs, RASStationInterface Property

Syntax DisconnectAfterSecs = _Long

Description This property allows you to set how long the connection must hold before retrying again after the previous attempt to connect failed.

Warning! This property is also available in the Communication Drivers' Basic Script interface. Even though both properties have the same function, they will not get mixed up with each other as they to are used in two different contexts.

Parameter	Description
None	None

Result Long

Example:

```
   Dim RasObj As RASStationInterface
   Dim NetwObj As NetworkClientCmd
Public Sub Click()
        If NetwObj Is Nothing Then Set NetwObj  = GetNetworkClient
        If RasObj  Is Nothing And Not NetwObj Is Nothing Then Set RasObj  =
        NetwObj.GetRASStation("FirstRAS")
        If Not RasObj Is Nothing Then
                RasObj.DisconnectAfterSecs = 30
        End If
End Sub
```

ShowConnectionDlg, RASStationInterface Property

Syntax ShowConnectionDlg = _Boolean

Description When this property is enabled a window will appear showing its status.

Warning! This property is also available in the Communication Drivers' Basic Script interface. Even though both properties have the same function, they will not get mixed up with each other as they to are used in two different contexts.

Parameter	Description
None	None

Result Boolean

Example:

```
   Dim RasObj As RASStationInterface
   Dim NetwObj As NetworkClientCmd
Public Sub Click()
        If NetwObj Is Nothing Then Set NetwObj  = GetNetworkClient
        If RasObj  Is Nothing And Not NetwObj Is Nothing Then Set RasObj  =
        NetwObj.GetRASStation("FirstRAS")
        If Not RasObj Is Nothing Then
                RasObj.ShowConnectionDlg = True
        End If
End Sub
```

StartConnectionTime, RASStationInterface Property

Syntax StartConnectionTime = _Date

Description This property returns the time and date in which the connection was made.

Warning! This property is also available in the Communication Drivers' Basic Script interface. Even though both properties have the same function, they will not get mixed up with each other as they to are used in two different contexts.

Parameter	Description
None	None

Result Date

Example:

```
   Dim RasObj As RASStationInterface
   Dim NetwObj As NetworkClientCmd
Public Sub Click()
        If NetwObj Is Nothing Then Set NetwObj  = GetNetworkClient
        If RasObj  Is Nothing And Not NetwObj Is Nothing Then Set RasObj  =
        NetwObj.GetRASStation("FirstRAS")
        If Not RasObj Is Nothing Then
                Debug.Print RasObj.StartConnectionTime
        End If
End Sub
```

TotalConnectionTime, RASStationInterface Property

Syntax TotalConnectionTime = _Date

Description This property returns the number of total connection time.

Warning! This property is also available in the Communication Drivers' Basic Script interface. Even though both properties have the same function, they will not get mixed up with each other as they to are used in two different contexts.

Parameter	Description
None	None

Result Date

Example:

```
   Dim RasObj As RASStationInterface
```

```
   Dim NetwObj As NetworkClientCmd
Public Sub Click()
        If NetwObj Is Nothing Then Set NetwObj  = GetNetworkClient
        If RasObj  Is Nothing And Not NetwObj Is Nothing Then Set RasObj  =
        NetwObj.GetRASStation("FirstRAS")
        If Not RasObj Is Nothing Then
                Debug.Print RasObj.TotalConnectionTime
        End If
End Sub
```

UserName, RASStationInterface Property

Syntax UserName = _String

Description This property allows you to insert the user name to be used for client station authenitication by Server station. In this case the user must be one who is recognized by the OS and therefore not necessarily a server project user, but one declared in the OS.
If the "Connection" property has been compiled, this field may be left blank.

Warning! This property is also available in the Communication Drivers' Basic Script interface. Even though both properties have the same function, they will not get mixed up with each other as they to are used in two different contexts.

Parameter	Description
None	None

Result String

Example:

```
   Dim RasObj As RASStationInterface
   Dim NetwObj As NetworkClientCmd
Public Sub Click()
        If NetwObj Is Nothing Then Set NetwObj  = GetNetworkClient
        If RasObj  Is Nothing And Not NetwObj Is Nothing Then Set RasObj  =
        NetwObj.GetRASStation("FirstRAS")
        If Not RasObj Is Nothing Then
                Debug.Print RasObj.PhoneBookEntry
        End If
End Sub
```

48. RecipeWndCmdTarget

OnActivateRecipe, RecipeWndCmdTarget Event

Description Event notified each time the "Activate" or the corresponding accelerator key is pressed.

Parameter	Description
bRet As Boolean	Enabling upon status change.

OnCopyRecipe, RecipeWndCmdTarget Event

Description Event notified each time the "Copy" or the corresponding accelerator key is pressed.

Parameter	Description
bRet As Boolean	Enabling upon status change.

OnDeleteRecipe, RecipeWndCmdTarget Event

Description Event notified each time the "Delete" or corresponding accelerator key is pressed.

Parameter	Description
bRet As Boolean	Enabling upon status change.

OnExportRecipe, RecipeWndCmdTarget Event

Description Event notified each time the "Export" key or corresponding accelerator key is pressed.

Parameter	Description
bRet As Boolean	Enabling upon status change.

OnImportRecipe, RecipeWndCmdTarget Event

Description Event notified each time the "Import" or corresponding accelerator key is pressed.

Parameter	Description
bRet As Boolean	Enabling upon status change.

OnPasteRecipe, RecipeWndCmdTarget Event

Description Event notified each time the "Paste" key or corresponding accelerator key is pressed.

Parameter	Description
bRet As Boolean	Enabling upon status change.

OnPrintRecipe, RecipeWndCmdTarget Event

Description Event notified each time the "Print" button is pressed or the corresponding accelerator key is used.

Parameter	Description
bRet As Boolean	report print enabling.

OnReadRecipe, RecipeWndCmdTarget Event

Description Event notified each time the "Read" key or corresponding accelerator key is pressed.

Parameter	Description
bRet As Boolean	Enabling upon status change.

OnRecipeIndexChanged, RecipeWndCmdTarget Event

Description Event notified each time the recipe selected from the drop-down list is changed.

Parameter	Description
ChangedRecipeIndex As String	Returns the name of the selected recipe.

OnRefreshRecipe, RecipeWndCmdTarget Event

Description Event notified each time the "Refresh" key or corresponding accelerator key is pressed.

Parameter	Description
bRet As Boolean	Enabling upon status change.

OnSaveRecipe, RecipeWndCmdTarget Event

Description Event notified each time the "Save" key or corresponding accelerator key is pressed.

Parameter	Description
bRet As Boolean	Enabling upon status change.

EditLayout, RecipeWndCmdTarget Function

Syntax EditLayout()

Description This function opens the configuration window for the fields to be displayed in the recipe Window.

This function will only be executed if the "Show Control Window" property has been enabled in the Window object. Otherwise the "Field Choice" window will not open and this function will return to the "False" value.

Parameter	Description
None	None

Result Boolean

Example:

```
Dim objRecipe As RecipeWndCmdTarget
Public Sub Click()
```

```
        Debug.Print objRecipe.EditLayout
End Sub
Public Sub SymbolLoading()
        Set objRecipe=
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
End Sub
```

ExportRecipeToCSV, RecipeWndCmdTarget Function

Syntax ExportRecipeToCSV(_lpszFileName)

Description This function exports grid values to the file specified in csv. format. The separator used for the various grid elements is the one set in the same object's property.
Using this function as described below, you will be able to customize predefined behaviour of the object's "Export" button.
The return value informs whether operation was a success or not.

Parameter	Description
lpszFileName As String	Name of file in which data must be exported. This name must be comprised of the path and file extension. In cases where an empty string is specified, the explorer window from the Windows resources will appear requesting user to indicate which file is to be used.

Result Boolean

Example:

```
Public Sub OnExportRecipe(ByRef bRet As Boolean)
        bRet = False

        Dim bResult As Boolean
        Dim sFileName As String

        sFileName = GetDataLoggerRecipePath & Recipe & ".csv"
        bResult = ExportRecipeToCSV(sFileName)

        If bResult Then
                MsgBox "Recipe exported successfully!"
        Else
                MsgBox "Error on exporting the Recipe!"
        End If
End Sub
```

ImportRecipeFromCSV, RecipeWndCmdTarget Function

Syntax ImportRecipeFromCSV (_lpszFileName)

Description This function imports grid values from the file specified in csv. format. The separator used for the various grid elements is the one set in the same object's property.
Using this function as described below, you will be able to customize predefined

behaviour of the object's "Import" button.
The return value informs whether operation was a success or not.

Parameter	Description
lpszFileName As String	Name of file in which data must be imported. This name must be comprised of the path and file extension. In cases where an empty string is specified, the explorer window from the Windows resources will appear requesting user to indicate which file is to be used.

Result Boolean

Example:

```
Public Sub OnImportRecipe(ByRef bRet As Boolean)
        bRet = False

        Dim bResult As Boolean
        Dim sFileName As String

        sFileName = GetDataLoggerRecipePath & Recipe & ".csv"
        bResult = ImportRecipeFromCSV (sFileName)

        If bResult Then
                MsgBox "Recipe imported successfully!"
        Else
                MsgBox "Error on importing the Recipe!"
        End If
End Sub
```

LoadExtSettings, RecipeWndCmdTarget Function

Syntax LoadExtSettings

Description This function allows you to load the object's setting from the relative external setting file. This file can be specified in the "Settings File" property during design mode or by using the "ExtSettingsFile" interface property. This extension provided for this file is ".SXML".

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
Dim objSymbol As RecipeWndCmdTarget
Set objSymbol = GetSynopticObject.GetSubObject("TestObject").GetObjectInterface
If objSymbol Is Nothing Then Exit Sub
objSymbol.ExtSettingsFile = "test.sxml"
objSymbol.LoadExtSettings
Set objSymbol = Nothing
```

```
End Sub
```

RecalcLayout, RecipeWndCmdTarget Function

Syntax RecalcLayout()

Description This function updates the object's graphics. This function must be used after modifying properties involving the object's graphical impact aspects.

Parameter	Description
None	None

Result Boolean

Example:

```
Dim objRecipe As RecipeWndCmdTarget
Public Sub Click()
        objRecipe.RecalcLayout
End Sub
Public Sub SymbolLoading()
        Set objRecipe=
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
End Sub
```

Reconnect, RecipeWndCmdTarget Function

Syntax Reconnect()

Description This function executes a reconnection to the database according to the recipe Resource associated to the display window and updates data by re-reading the corresponding database table. This function must be called to reload data if the recipe Resource to the associated to the window changes or is connected to a network Server through the "NetworkServerName" property.

Parameter	Description
None	None

Result Boolean

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe.Recipe = "Recipe1"
        objRecipe.Reconnect
        Set objRecipe = Nothing
```

```
End Sub
```

Refresh, RecipeWndCmdTarget Function

Syntax Refresh()

Description This function executes a refresh of data displayed in the object by reloading data currently in the database. However, if the recipe Resource associated to the window has been changed you will need to call the Reconnect method.

Parameter	Description
None	None

Result Boolean

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe.Refresh
        Set objRecipe = Nothing
End Sub
```

SaveExtSettings, RecipeWndCmdTarget Function

Syntax SaveExtSettings

Description This function allows you to save the the object's configuration in the relating external settings file. This file can be specified in design mode in the "Ext. File Settings property", or using the "ExtSettingsFile" interface property. The extension to use for this file is ".SXML".

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
Dim objSymbol As RecipeWndCmdTarget
Set objSymbol = GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
If objSymbol Is Nothing Then Exit Sub
objSymbol.ExtSettingsFile = "test.sxml"
objSymbol.SaveExtSettings
Set objSymbol = Nothing
End Sub
```

ActivateBtnText, RecipeWndCmdTarget Property

Syntax ActivateBtnText = _String

Description This property sets or resets the text to be displayed in the "Activate" button (if one exists) of the recipe window. The default text will be used if text is not specified. Call the RecalcLayout method to put modification into effect.

Parameter	Description
None	None

Result String

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe.ActivateBtnText = "Activate recipe"
        objRecipe.RecalcLayout
        Set objRecipe = Nothing
End Sub
```

ActivateMessage, RecipeWndCmdTarget Property

Syntax ActivateMessage = _String

Description This property sets or returns the text to be displayed as the string for the Message Box requesting an OK and which opens when the "activate" recipe command is prompted. The Message Box will not display and the command will activate without asking for a confirm if this property is set with an empty string. A string ID can also be inserted.

Parameter	Description
None	None

Result String

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe.ActivateMessage = "Do you want to activate the selected Recipe?"
        Set objRecipe = Nothing
End Sub
```

AutoLayout, RecipeWndCmdTarget Property

Syntax AutoLayout = _Boolean

Description Enabling this property, will set the list layout automatically. This means that the table columns will automatically be resized so that they all fit in and are visible within the DataLogger/Recipe Window area. Disabling this property will make the columns appear with their original sizes set in design mode and therefore the last columns on the right may not fit within window unless the horizontal scroll bar is activated to reach them.

Parameter	Description
None	None

Result Boolean

Example:

```
Dim objRecipe As RecipeWndCmdTarget
Public Sub Click()
        Debug.Print objRecipe.AutoLayout
End Sub
Public Sub SymbolLoading()
        Set objRecipe=
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
End Sub
```

ButtonPos, RecipeWndCmdTarget Property

Syntax ButtonPos = _Integer

Description This setting returns the position in which the buttons must appear in the data display window.

Positions may be:
0 = left
1 = top
2 = right
3 = bottom

Parameter	Description
None	None

Result Integer

Example:

```
Dim objRecipe As ButtonPos, RecipeWndCmdTarget Property
Public Sub Click()
        If Not objRecipe Is Nothing Then
                MsgBox "objRecipe 's ButtonPos is " & objRecipe
                .ButtonPos,vbInformation,GetProjectTitle
                objRecipe.ButtonPos = 2
                objRecipe.RecalcLayout
```

```
        Else
                MsgBox "objRecipe is nothing",vbInformation,GetProjectTitle
        End If
End Sub
Public Sub SymbolLoading()
        Set                                objRecipe                                =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
End Sub
```

ButtonSize, RecipeWndCmdTarget Property

Syntax ButtonSize = _Integer

Description This setting returns the size of the buttons which are to bed displayed in the data display window.

Possible sizes are:
0 = small
1 = medium
2 = large

Parameter	Description
None	None

Result Integer

Example:

```
Dim objRecipe As ButtonSize, RecipeWndCmdTarget Property
Public Sub Click()
        If Not objRecipe Is Nothing Then
                MsgBox "objRecipe 's ButtonSize is " &
                objRecipe.ButtonSize,vbInformation,GetProjectTitle
                objRecipe.ButtonSize = 2
                objRecipe.RecalcLayout
        Else
                MsgBox "objRecipe is nothing",vbInformation,GetProjectTitle
        End If
End Sub
Public Sub SymbolLoading()
        Set                                objRecipe                                =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
End Sub
```

Clickable, RecipeWndCmdTarget Property

Syntax Clickable = _Boolean

Description This property lets you specify whether the user can interact with the Display window or not. Setting this property to False will impede the use of the mouse and keyboard to manage display control. In this case, it will not be possible to put columns into the desired order by using the commands shown in the windows. Please refer to the corresponding help on this matter.

Parameter	Description
None	None

Result Boolean

Example:

```
Dim objRecipe As Clickable, RecipeWndCmdTarget Property
Public Sub Click()
        Debug.Print objRecipe.Project
End Sub
Public Sub SymbolLoading()
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
End Sub
```

CopyBtnText, RecipeWndCmdTarget Property

Syntax CopyBtnText = _String

Description This property set or returned the text to be displayed in the "Copy" button (if one exists) of the recipe's display window. the default text will be used if no text is entered. To put this modification into effect you will need to call the RecalcLayout method.

Parameter	Description
None	None

Result String

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe.CopyBtnText = "Copy recipe"
        objRecipe.RecalcLayout
        Set objRecipe = Nothing
End Sub
```

CurrentRecipeVariable, RecipeWndCmdTarget Property

Syntax CurrentRecipeVariable = _String

Description This property sets or returns the name of the project's variable in which the name of the currently active recipe will be inserted. Corresponds to the Recipe Manager window's "Current Recipe Var." property.

Parameter	Description
None	None

Result String

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Dim sText as String
        GetVariableNameFromList(sText)
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe.CurrentRecipeVariable = sText
        Set objRecipe = Nothing
End Sub
```

DeleteBtnText, RecipeWndCmdTarget Property

Syntax DeleteBtnText = _String

Description This property set or returns the text to be displayed in the "Delete" button (if one exists) in the Recipe window. The default text will be used if no other text is specified. To put this modification into effect you will need to call the RecalcLayout method.

Parameter	Description
None	None

Result String

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe.DeleteBtnText = "Delete recipe"
        objRecipe.RecalcLayout
        Set objRecipe = Nothing
End Sub
```

DeleteMessage, RecipeWndCmdTarget Property

Syntax DeleteMessage = _String

Description This property sets or returns the text to be displayed as the string for the Message Box requesting a confirm and which opens when prompting the recipe "Delete" command. The Message Box will not display when setting

this property with an empty string and the command will be activated without asking for a confirm. String ID may also be inserted.

Parameter	Description
None	None

Result String

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe.DeleteMessage = "Do you want to delete the selected Recipe?"
        Set objRecipe = Nothing
End Sub
```

ErrorString, RecipeWndCmdTarget Property

Syntax ErrorString = _String

Description This property sets or reset the text to be displayed as the error string while setting valued in the grid's "Value" column if editable. When setting this string with and empty string, a beep will sound. Min and Max limest can be displayed in the error string for numeric variables using the "%d" notation instead of the value (see example).

Parameter	Description
None	None

Result String

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe.ErrorString = "Value out of range:  min = %d, max = %d"
        Set objRecipe = Nothing
End Sub
```

ExportBtnText, RecipeWndCmdTarget Property

Syntax ExportBtnText = _String

Description This property sets or returns the text to be displayed in the recipe window's "Export" button (if exists). The default text will be used if no test is specified. To put this modification into effect you will need to call

the RecalcLayout method.

Parameter	Description
None	None

Result String

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe.ExportBtnText = "Export recipe"
        objRecipe.RecalcLayout
        Set objRecipe = Nothing
End Sub
```

ExtSettingsFile, RecipeWndCmdTarget Property

Syntax ExtSettingsFile = _String

Description This property sets or returns the external configuration file for the referenced object. This file can also be specified in design mode in the object's 'Ext. File Settings' property. The extension provided for this file is ".SXML".

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim objSymbol AsExtSettingsFile, RecipeWndCmdTarget Property
        Set objSymbol =
        GetSynopticObject.GetSubObject("TestObject").GetObjectInterface
        If objSymbol Is Nothing Then Exit Sub
        objSymbol.ExtSettingsFile = "test.sxml"
        objSymbol.SaveExtSettings
        Set objSymbol= Nothing
End Sub
```

GraphicButtons, RecipeWndCmdTarget Property

Syntax GraphicButtons = _Boolean

Description When Enabling this property, the Recipe Window buttons are drawn using an icon instead of text. The text will instead be displayed as a tooltip when positioning the mouse on top of the button.

The tooltip is not managed in Windows CE versions.

This property is not managed by the 'Alarm Banner' object.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Click()
        GraphicButtons = True
        RecalcLayout
End Sub
```

ImpExpSeparator, RecipeWndCmdTarget Property

Syntax ImpExpSeparator = _Integer

Description This property sets or resets the ASCII code of the character used as the column separator in the recipe's export/import file.

Parameter	Description
None	None

Result Integer

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe.ImpExpSeparator = Asc(",")
        Set objRecipe = Nothing
End Sub
```

ImportBtnText, RecipeWndCmdTarget Property

Syntax ImportBtnText = _String

Description This property sets or returns the text to be displayed in the Recipe Window's "Import" button (if one exists). The defualt text will be used if no text is specified. To put this modification into effect you will need to call the RecalcLayout method.

Parameter	Description
None	None

Result String

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe.ImportBtnText = "Import recipe"
        objRecipe.RecalcLayout
        Set objRecipe = Nothing
End Sub
```

NetworkBackupServerName, RecipeWndCmdTarget Property

Syntax NetworkBackupServerName = _String

Description This property sets or returns the name of any Network Backup Server used for getting data to display in the Recipe Manager Window when the primary server, the one set in the 'NetowrkServerName'property is in timeout.

To display data from a Server, the recipe must also be present in the Client project, so that the Database structure can be retrieved. However, the Recipe can only be created as structure type in the Client project, therefore without associating any variables to columns.

Parameter	Description
None	None

Result String

Example:

```
Dim objRecipeWnd As RecipeWndCmdTarget
Public Sub Click()
        Debug.Print objRecipeWnd.NetworkBackupServerName
End Sub
Public Sub SymbolLoading()
        Set objRecipeWnd =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
End Sub
```

NetworkServerName, RecipeWndCmdTarget Property

Syntax NetworkServerName = _String

Description This property returns the name of any Network Server where data is to be retrieved for displaying in the Recipe Manager window.

To display data from a Server, the Recipe will also need to be in the Client project in order to retrieve the Database structure. However, the recipe can only be created as structure type in the Client project without associating variables to its columns.

Parameter	Description
None	None

Result String

Example:

```
Dim objRecipe As RecipeWndCmdTarget
Public Sub Click()
        Debug.Print objRecipe.NetworkServerName
End Sub
Public Sub SymbolLoading()
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
End Sub
```

PasteBtnText, RecipeWndCmdTarget Property

Syntax PasteBtnText = _String

Description This property sets or returns the text to be displayed in the Recipe window's "Paste" button (if one exists). The default text will be used if one isn't entered. Call the RecalcLayout method to put modification into effect.

Parameter	Description
None	None

Result String

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
```

```
        objRecipe.PasteBtnText = "Paste recipe"
        objRecipe.RecalcLayout
        Set objRecipe = Nothing
End Sub
```

Project, RecipeWndCmdTarget Property

Syntax Project = _String

Description This property sets or returns the name of the child project from which data is retrieved for displaying. The current project will be used if this field is left empty.

Only the name of an eventual child project in the current project can be entered in this filed.

Parameter	Description
None	None

Result String

Example:

```
Dim objRecipe As Project, RecipeWndCmdTarget Property
Public Sub Click()
        Debug.Print objRecipe.Project
End Sub
Public Sub SymbolLoading()
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
End Sub
```

PromptPad, RecipeWndCmdTarget Property

Syntax PromptPad = _Boolean

Description This property enables or disables the option to display the Numeric or Alphanumeric Pad when the user enters one of the grid's "Value" column cells, if editable, in editing mode (with mouse click or keyboard key). The Numeric Pad shows when the variable is numeric type, otherwise the Alphanumeric Pad will show.

Parameter	Description
None	None

Result Boolean

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
```

```
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe.PromptPad = Not objRecipe.PromptPad
        Set objRecipe = Nothing
End Sub
```

PrintBtnText, RecipeWndCmdTarget Property

Syntax PrintBtnText = _String

Description This property sets or returns the text that is to display in the Recipe window's "Print" button (if present). If text is not specified, the default text will be used instead. To put modification into effect you will need to call the RecalcLayout method.

Parameter	Description
None	None

Result String

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe. PrintBtnText = "Print recipe"
        objRecipe.RecalcLayout
        Set objRecipe = Nothing
End Sub
```

ReadBtnText, RecipeWndCmdTarget Property

Syntax ReadBtnText = _String

Description This property sets or returns the text to be displayed in the Recipe Window's "Read" button (if one exists). In order to put this modification into effect you will need to call the RecalcLayout method.

Parameter	Description
None	None

Result String

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe.ReadBtnText = "Read recipe"
        objRecipe.RecalcLayout
```

```
        Set objRecipe = Nothing
End Sub
```

Recipe, RecipeWndCmdTarget Property

Syntax Recipe = _String

Description This property sets or resets the name of the Recipe Resource associated to the display window. If you modify the Recipe Resource associated to the display window and you wish to display data relating to a new recipe Resource you will need to call the Reconnect method.

Parameter	Description
None	None

Result String

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        Debug.Print objRecipe.Recipe
        objRecipe.Recipe = "Recipe1"
        objRecipe.Reconnect
        Set objRecipe = Nothing
End Sub
```

RefreshBtnText, RecipeWndCmdTarget Property

Syntax RefreshBtnText = _String

Description This property sets or returns the text to be displayed in the Recipe window's 'Refresh' button (if one exists). The default text will be used If no text is speciified. In order to put this modification into effect you must call the RecalcLayout method.

Parameter	Description
None	None

Result String

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe.RefreshBtnText = "Refresh recipe"
```

```
        objRecipe.RecalcLayout
        Set objRecipe = Nothing
End Sub
```

SaveBtnText, RecipeWndCmdTarget Property

Syntax SaveBtnText = _String

Description This property sets or returns the text to be displayed in the recipe window's "Save" button (if one exists). If this text is not specified the default text will be use instead. To put this modification into effect you will need to call the RecalcLayout method.

Parameter	Description
None	None

Result String

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe.SaveBtnText = "Save recipe"
        objRecipe.RecalcLayout
        Set objRecipe = Nothing
End Sub
```

SaveMessage, RecipeWndCmdTarget Property

Syntax SaveMessage = _String

Description This property sets or returns the text to be displayed as the string for Message Box requesting a confirm and which opened when the recipe "save" command is activated. The Message Box will not show if this property is set with an empty string and the command will go ahead with requiring user confirmation. A string ID can also be entered here.

Parameter	Description
None	None

Result String

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe.SaveMessage = "Do you want to save the selected Recipe?"
```

```
        Set objRecipe = Nothing
End Sub
```

ShowActivateBtn, RecipeWndCmdTarget Property

Syntax ShowActivateBtn = _Boolean

Description This property permits the "Active" button to be shown or hidden in the recipe window. To put this modification into effect, you will need to call the RecalcLayout method.

Parameter	Description
None	None

Result Boolean

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe.ShowActivateBtn = Not objRecipe.ShowActivateBtn
        objRecipe.RecalcLayout
        Set objRecipe = Nothing
End Sub
```

ShowCopyBtn, RecipeWndCmdTarget Property

Syntax ShowCopyBtn = _Boolean

Description This property allows the "Copy" button to be shown or hidden in the recipe window. To put this modification into effect you will need to call the RecalcLayout method.

Parameter	Description
None	None

Result Boolean

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe.ShowCopyBtn = Not objRecipe.ShowCopyBtn
        objRecipe.RecalcLayout
        Set objRecipe = Nothing
```

```
End Sub
```

ShowDeleteBtn, RecipeWndCmdTarget Property

Syntax ShowDeleteBtn = _Boolean

Description This property allows the "Delete" button to be shown or hidden in the recipe window. To put this modification into effect you will need to call the RecalcLayout method.

Parameter	Description
None	None

Result Boolean

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe.ShowDeleteBtn = Not objRecipe.ShowActivateBtn
        objRecipe.RecalcLayout
        Set objRecipe = Nothing
End Sub
```

ShowExportBtn, RecipeWndCmdTarget Property

Syntax ShowExportBtn = _Boolean

Description This property allows the "Export" button to be shown or hidden in the recipe window. To put this modification into effect you will need to call the RecalcLayout method.

Parameter	Description
None	None

Result Boolean

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe.ShowExportBtn = Not objRecipe.ShowExportBtn
        objRecipe.RecalcLayout
```

```
        Set objRecipe = Nothing
End Sub
```

ShowImportBtn, RecipeWndCmdTarget Property

Syntax ShowImportBtn = _Boolean

Description This property allows the "Import" button to be shown or hidden in the recipe window. To put this modification into effect you will need to call the RecalcLayout method.

Parameter	Description
None	None

Result Boolean

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe.ShowImportBtn = Not objRecipe.ShowImportBtn
        objRecipe.RecalcLayout
        Set objRecipe = Nothing
End Sub
```

ShowPasteBtn, RecipeWndCmdTarget Property

Syntax ShowPasteBtn = _Boolean

Description This property allows the "Paste" button to be shown or hidden in the recipe window. To put this modification into effect you will need to call the RecalcLayout method.

Parameter	Description
None	None

Result Boolean

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe.ShowPasteBtn = Not objRecipe.ShowPasteBtn
```

```
        objRecipe.RecalcLayout
        Set objRecipe = Nothing
End Sub
```

ShowPrintBtn, RecipeWndCmdTarget Property

Syntax ShowPrintBtn= _Boolean

Description This button shows or hides the "Print" button in the recipe window. The RecalcLayout method needs to be called in order to put this modification into effect.

Parameter	Description
None	None

Result Boolean

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe. ShowPrintBtn = Not objRecipe. ShowPrintBtn
        objRecipe.RecalcLayout
        Set objRecipe = Nothing
End Sub
```

ShowReadBtn, RecipeWndCmdTarget Property

Syntax ShowReadBtn = _Boolean

Description This property allows the "Read" button to be shown or hidden in the recipe window. To put this modification into effect you will need to call the RecalcLayout method.

Parameter	Description
None	None

Result Boolean

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
```

```
        objRecipe.ShowReadBtn = Not objRecipe.ShowReadBtn
        objRecipe.RecalcLayout
        Set objRecipe = Nothing
End Sub
```

ShowRefreshBtn, RecipeWndCmdTarget Property

Syntax ShowRefreshBtn = _Boolean

Description This property allows the "Refresh" button to be shown or hidden in the recipe window. To put this modification into effect you will need to call the RecalcLayout method.

Parameter	Description
None	None

Result Boolean

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe.ShowRefreshBtn = Not objRecipe.ShowRefreshBtn
        objRecipe.RecalcLayout
        Set objRecipe = Nothing
End Sub
```

ShowSaveBtn, RecipeWndCmdTarget Property

Syntax ShowSaveBtn = _Boolean

Description This property allows the "Save" button to be shown or hidden in the recipe window. To put this modification into effect you will need to call the RecalcLayout method.

Parameter	Description
None	None

Result Boolean

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
```

```
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe.ShowSaveBtn = Not objRecipe.ShowSaveBtn
        objRecipe.RecalcLayout
        Set objRecipe = Nothing
End Sub
```

SubItemDescription, RecipeWndCmdTarget Property

Syntax SubItemDescription= _String

Description This property sets or resets the text to be displayed as the Recipe Window "Description" column's title (if one exists). The default text will be used if no text is entered. To put this modification into effect you will need to call the RecalcLayout method.

Parameter	Description
None	None

Result String

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe.SubItemDescription = "Description"
        objRecipe.RecalcLayout
        Set objRecipe = Nothing
End Sub
```

SubItemDescriptionPos, RecipeWndCmdTarget Property

Syntax SubItemDescriptionPos= _String

Description This property sets or returns the position of the "Descripton" column within Recipe Manager window. When setting a new value, the other columns will be automatically re-positioned in the window layout. In addition when setting the "-1", the column will be hidden. The "0" value is used to indicate position of the first column on the left in the window.

Parameter	Description
None	None

Result Integer

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe.SubItemDescriptionPos = 1
        Set objRecipe = Nothing
End Sub
```

SubItemDescriptionWidth, RecipeWndCmdTarget Property

Syntax SubItemDescriptionWidth = _Integer

Description This property sets or returns the column's size in pixels within the Recipe Window. The value -1 corresponds to the undisplayed column. The value 0 corresponds to the first displayed column. When changing the sizes of columns in the window you will need to call the RecalcLayout method to put changes into effect.

Parameter	Description
None	None

Result Integer

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe.SubItemDescriptionWidth = 100
        objRecipe.RecalcLayout
        Set objRecipe = Nothing
End Sub
```

SubItemMax, RecipeWndCmdTarget Property

Syntax SubItemMax = _String

Description This property sets or resets the text to be displayed as the Recipe Window "Max" column's title (if one exists). The default text will be used if no text is entered. To put this modification into effect you will need to call the RecalcLayout method.

Parameter	Description
None	None

Result String

Example:

```
Public sub Click()
```

```
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe.SubItemMax = "Max"
        objRecipe.RecalcLayout
        Set objRecipe = Nothing
End Sub
```

SubItemMaxPos, RecipeWndCmdTarget Property

Syntax SubItemMaxPos = _Integer

Description This property sets or returns the position of the "Max" column within Recipe Manager window. When setting a new value, the other columns will be automatically re-positioned in the window layout. In addition when setting the "-1", the column will be hidden. The "0" value is used to indicate position of the first column on the left in the window.

Parameter	Description
None	None

Result Integer

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe.SubItemMaxPos = 5
        Set objRecipe = Nothing
End Sub
```

SubItemMaxWidth, RecipeWndCmdTarget Property

Syntax SubItemMaxWidth = _Integer

Description This property sets or returns the column's size in pixels within the Recipe Window. The value -1 corresponds to the undisplayed column. The value 0 corresponds to the first displayed column. When changing the sizes of columns in the window you will need to call the RecalcLayout method to put changes into effect.

Parameter	Description
None	None

Result Integer

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe.SubItemMaxWidth = 100
        objRecipe.RecalcLayout
        Set objRecipe = Nothing
End Sub
```

SubItemMin, RecipeWndCmdTarget Property

Syntax SubItemMin = _String

Description This property sets or resets the text to be displayed as the Recipe Window "Min" column's title (if one exists). The default text will be used if no text is entered. To put this modification into effect you will need to call the RecalcLayout method.

Parameter	Description
None	None

Result String

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe.SubItemMin = "Min"
        objRecipe.RecalcLayout
        Set objRecipe = Nothing
End Sub
```

SubItemMinPos, RecipeWndCmdTarget Property

Syntax SubItemMinPos = _Integer

Description This property sets or returns the position of the "Min" column within Recipe Manager window. When setting a new value, the other columns will be automatically re-positioned in the window layout. In addition when setting the "-1", the column will be hidden. The "0" value is used to indicate position of the first column on the left in the window.

Parameter	Description
None	None

Result Integer

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe.SubItemMinPos = 4
        Set objRecipe = Nothing
End Sub
```

SubItemMinWidth, RecipeWndCmdTarget Property

Syntax SubItemMinWidth = _Integer

Description This property sets or returns the column's size in pixels within the Recipe Window. The value -1 corresponds to the undisplayed column. The value 0 corresponds to the first displayed column. When changing the sizes of columns in the window you will need to call the RecalcLayout method to put changes into effect.

Parameter	Description
None	None

Result Integer

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe.SubItemMinWidth = 100
        objRecipe.RecalcLayout
        Set objRecipe = Nothing
End Sub
```

SubItemUnits, RecipeWndCmdTarget Property

Syntax SubItemUnits = _String

Description This property sets or resets the text to be displayed as the Recipe Window "Units" column's title (if one exists). The default text will be used if no text is entered. To put this modification into effect you will need to call the RecalcLayout method.

Parameter	Description
None	None

Result String

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe.SubItemUnits = "Units"
        objRecipe.RecalcLayout
        Set objRecipe = Nothing
End Sub
```

SubItemUnitsPos, RecipeWndCmdTarget Property

Syntax SubItemUnitsPos = _Integer

Description This property sets or returns the position of the "Units" column within theRecipe Manager window. When setting a new value, the other columns will be automatically re-positioned in the window layout. In addition when setting the "-1", the column will be hidden. The "0" value is used to indicate position of the first column on the left in the window.

Parameter	Description
None	None

Result Integer

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe.SubItemUnitsPos = 3
        Set objRecipe = Nothing
End Sub
```

SubItemUnitsWidth, RecipeWndCmdTarget Property

Syntax SubItemUnitsWidth = _Integer

Description This property sets or returns the column's size in pixels within the Recipe Window. The value -1 corresponds to the undisplayed column. The value 0 corresponds to the first displayed column. When changing the sizes of columns in the window you will need to call the RecalcLayout method to put changes into effect.

Parameter	Description

None	None

Result Integer

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe.SubItemUnitsWidth = 100
        objRecipe.RecalcLayout
        Set objRecipe = Nothing
End Sub
```

SubItemValue, RecipeWndCmdTarget Property

Syntax SubItemValue = _String

Description This property sets or resets the text to be displayed as the Recipe Window "Value" column's title (if one exists). The default text will be used if no text is entered. To put this modification into effect you will need to call the RecalcLayout method.

Parameter	Description
None	None

Result String

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe.SubItemValue = "Value"
        objRecipe.RecalcLayout
        Set objRecipe = Nothing
End Sub
```

SubItemValuePos, RecipeWndCmdTarget Property

Syntax SubItemValuePos = _Integer

Description This property sets or returns the position of the "Value" column within the Recipe Manager window. When setting a new value, the other columns will be automatically re-positioned in the window layout. In addition when setting the "-1", the column will be hidden. The "0" value is used to indicate position of the first column on the left in the window.

Parameter	Description
None	None

Result Integer

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe.SubItemValuePos = 2
        Set objRecipe = Nothing
End Sub
```

SubItemValueWidth, RecipeWndCmdTarget Property

Syntax SubItemValueWidth = _Integer

Description This property sets or returns the column's size in pixels within the Recipe Window. The value -1 corresponds to the undisplayed column. The value 0 corresponds to the first displayed column. When changing the sizes of columns in the window you will need to call the RecalcLayout method to put changes into effect.

Parameter	Description
None	None

Result Integer

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe.SubItemValueWidth = 100
        objRecipe.RecalcLayout
        Set objRecipe = Nothing
End Sub
```

SubItemVariable, RecipeWndCmdTarget Property

Syntax SubItemVariable = _String

Description This property sets or resets the text to be displayed as the Recipe Window "Variable" column's title (if one exists). The default text will be used if no text is entered. To put this modification into effect you will need to call the RecalcLayout method.

Parameter	Description
None	None

Result String

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe.SubItemVariable = "Variable"
        objRecipe.RecalcLayout
        Set objRecipe = Nothing
End Sub
```

SubItemVariablePos, RecipeWndCmdTarget Property

Syntax SubItemVariablePos = _Integer

Description This property sets or returns the position of the "Variable" column within the Recipe Manager window. When setting a new value, the other columns will be automatically re-positioned in the window layout. In addition when setting the "-1", the column will be hidden. The "0" value is used to indicate position of the first column on the left in the window.

Parameter	Description
None	None

Result Integer

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe.SubItemVariablePos = 0
        Set objRecipe = Nothing
End Sub
```

SubItemVariableWidth, RecipeWndCmdTarget Property

Syntax SubItemVariableWidth = _Integer

Description This property sets or returns the column's size in pixels within the Recipe Window. The value -1 corresponds to the undisplayed column. The value

0 corresponds to the first displayed column. When changing the sizes of columns in the window you will need to call the RecalcLayout method to put changes into effect.

Parameter	Description
None	None

Result Integer

Example:

```
Public sub Click()
        Dim objRecipe As RecipeWndCmdTarget
        Set objRecipe =
        GetSynopticObject.GetSubObject("RecipeWindow").GetObjectInterface
        objRecipe.SubItemVariableWidth = 100
        objRecipe.RecalcLayout
        Set objRecipe = Nothing
End Sub
```

49. ScalingCmdTarget

GetXMLSettings, ScalingCmdTarget Function

Syntax GetXMLSettings()

Description This function returns a string with the contents of the project's XML file relating to the referred scaling object.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim ScalObj As ScalingCmdTarget
        Set ScalObj = GetScaling("Scal1")
        If Not ScalObj Is Nothing Then
                MsgBox ScalObj.GetXMLSettings ,vbOkOnly,""
        End If
        Set ScalObj = Nothing
End Sub
```

Reinit, ScalingCmdTarget Function

Syntax Reinit()

Description This method re-initializes a Scaling object and is used in runtime to active any modifications done to values of its properties.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
        Dim ScalObj As ScalingCmdTarget
        Set ScalObj = GetScaling("Scal1")
        ScalObj.RawMaxValue = 10
        ScalObj.Reinit
        Set ScalObj = Nothing
End Sub
```

DeadBandValue, ScalingCmdTarget Property

Syntax DeadBandValue = _Double

Description This property sets or returns the "dead band" value in the conversion factor. The dead band establishes the value to which the scaled variable is set in cases where the value of the unscaled variable exists from the set conversion tolerance. The default value set by Movicon is "-1".

Parameter	Description
None	None

Result Double

Example:

```
Option Explicit
Public Sub Click()
        Dim ScalObj As ScalingCmdTarget
        Set ScalObj = GetScaling("Scal1")
        If Not ScalObj Is Nothing Then
                Debug.Print ScalObj.DeadBandValue
        End If
        Set ScalObj = Nothing
End Sub
```

Enabled, ScalingCmdTarget Property

Syntax Enabled = _Boolean

Description This property enables or disables the reference scaling object. When the value is left set to False, the conversion operations will not be executed.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
        Dim ScalObj As ScalingCmdTarget
        Set ScalObj = GetScaling("Scal1")
        If Not ScalObj Is Nothing Then
                Debug.Print ScalObj.Enabled
        End If
        Set ScalObj = Nothing
End Sub
```

Name, ScalingCmdTarget Property

Syntax Name = _String

Description This function returns a string with the name of the reference Scaling object.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim ScalObj As ScalingCmdTarget
        Set ScalObj = GetScaling("Scal1")
        If Not ScalObj Is Nothing Then
                MsgBox "Scaling Name Is " & ScalObj.Name,vbOkOnly,""
        End If
        Set ScalObj = Nothing
End Sub
```

RawMaxValue, ScalingCmdTarget Property

Syntax RawMaxValue = _Double

Description This property sets or returns the maximum value of the raw variable being the input value. The minimum and maximum scaled output value is calculated according to the minimum and maximum raw input value according to a linear function.

Parameter	Description
None	None

Result Double

Example:

```
Option Explicit
Public Sub Click()
        Dim ScalObj As ScalingCmdTarget
        Set ScalObj = GetScaling("Scal1")
        If Not ScalObj Is Nothing Then
                Debug.Print ScalObj.RawMaxValue
        End If
        Set ScalObj = Nothing
End Sub
```

RawMinValue, ScalingCmdTarget Property

Syntax RawMinValue = _Double

Description This property sets or returns the raw minimum value, being the input value. The minimum and maximum scaled output value is calculated according to the minimum and maximum raw input value according to a linear function.

Parameter	Description
None	None

Result Double

Example:

```
Option Explicit
Public Sub Click()
        Dim ScalObj As ScalingCmdTarget
        Set ScalObj = GetScaling("Scal1")
        If Not ScalObj Is Nothing Then
                Debug.Print ScalObj.RawMinValue
        End If
        Set ScalObj = Nothing
End Sub
```

RawVariableName, ScalingCmdTarget Property

Syntax RawVariableName = _String

Description This function returns a string with the name of the variable containing the raw value to be scaled.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim ScalObj As ScalingCmdTarget
        Set ScalObj = GetScaling("Scal1")
        If Not ScalObj Is Nothing Then
                MsgBox ScalObj.RawVariableName,vbOkOnly,""
        End If
        Set ScalObj = Nothing
End Sub
```

ScaledMaxValue, ScalingCmdTarget Property

Syntax ScaledMaxValue = _Double

Description This property sets or returns the maximum value of the scaled variable, being the output value corresponding to the real physical size.

Parameter	Description
None	None

Result Double

Example:

```
Option Explicit
Public Sub Click()
        Dim ScalObj As ScalingCmdTarget
        Set ScalObj = GetScaling("Scal1")
        If Not ScalObj Is Nothing Then
                Debug.Print ScalObj.ScaledMaxValue
        End If
        Set ScalObj = Nothing
End Sub
```

ScaledMinValue, ScalingCmdTarget Property

Syntax ScaledMinValue = _Double

Description This property sets or returns the minimum value of the scaled variable, being the output value corresponding to the real physical size.

Parameter	Description
None	None

Result Double

Example:

```
Option Explicit
Public Sub Click()
        Dim ScalObj As ScalingCmdTarget
        Set ScalObj = GetScaling("Scal1")
        If Not ScalObj Is Nothing Then
                Debug.Print ScalObj.ScaledMinValue
        End If
        Set ScalObj = Nothing
End Sub
```

ScaleVariableName, ScalingCmdTarget Property

Syntax ScaleVariableName = _String

Description This function returns a string with the name of the variable which will contain the scaled value, being the conversion calculation result based on the conversion factors set in the following described property.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim ScalObj As ScalingCmdTarget
        Set ScalObj = GetScaling("Scal1")
        If Not ScalObj Is Nothing Then
                MsgBox ScalObj.ScaleVariableName,vbOkOnly,""
        End If
        Set ScalObj = Nothing
End Sub
```

50. SchedulerCmdTarget

AddHoliday, SchedulerCmdTarget Function

Syntax AddHoliday(_dt)

Description This function allows a Holiday to be added to the Schedular object. The date is added only when not already done so otherwise this function returns "false". A "date" type parameter with day and month only must be passed to the function.

Parameter	Description
_dt as date	Holiday Date to be managed

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
        Dim SchedObj As SchedulerCmdTarget
        Set SchedObj = GetScheduler("Sched1")
        If Not SchedObj Is Nothing Then
            SchedObj.AddHoliday(now)
        End If
        Set SchedObj = Nothing
End Sub
```

GetHolidaysString, SchedulerCmdTarget Function

Syntax GetHolidaysString(_lpszSep)

Description This function returns a string divided by the pre-chosen separation character (function parameter) with the list of all the holiday dates set in the scheduler object. These dates are expressed in days and months only.

Parameter	Description
lpszSep as string	Separator character between the dates returned in the string from the function.

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim SchedObj As SchedulerCmdTarget
        Set SchedObj = GetScheduler("Sched1")
```

```
        If Not SchedObj Is Nothing Then
            MsgBox "Holidays Date = " &
            CStr(SchedObj.GetHolidaysString(",")),vbOkOnly,GetProjectTitle
        End If
        Set SchedObj = Nothing
End Sub
```

GetXMLSettings, SchedulerCmdTarget Function

Syntax GetXMLSettings()

Description This function returns a string with the contents of the project's XML file relating to the referenced scheduler object.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim SchedObj As SchedulerCmdTarget
        Set SchedObj = GetScheduler("Sched1")
        If Not SchedObj Is Nothing Then
            MsgBox SchedObj.GetXMLSettings ,vbOkOnly,""
        End If
        Set SchedObj = Nothing
End Sub
```

IsHoliday, SchedulerCmdTarget Function

Syntax IsHoliday(_dt)

Description This function allows you to verify whether a date has been set as a holiday in the scheduler object. You need to pass a "date" parameter to this function where only the day and month will be considered.

Parameter	Description
_dt as date	Date to be removed from the holiday list.

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
        Dim SchedObj As SchedulerCmdTarget
        Set SchedObj = GetScheduler("Sched1")
        If Not SchedObj Is Nothing Then
```

```
            MsgBox "Is Holiday = " & CStr(SchedObj.IsHoliday(Now)),vbOkOnly,GetProjectTitle
        End If
        Set SchedObj = Nothing
End Sub
```

RemoveHoliday, SchedulerCmdTarget Function

Syntax RemoveHoliday(_dt)

Description This function allows you to remove a Holiday from the scheduler object. The date is removed only if present, otherwise this function returns "false". You need to pass a "date" parameter to this function where only the day and month are considered.

Parameter	Description
_dt as date	Date to be removed from list of holidays.

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
        Dim SchedObj As SchedulerCmdTarget
        Set SchedObj = GetScheduler("Sched1")
        If Not SchedObj Is Nothing Then
            SchedObj.RemoveHoliday(now)
        End If
        Set SchedObj = Nothing
End Sub
```

Reset, SchedulerCmdTarget Function

Syntax Reset()

Description Resets the Scheduler's behaviour. Permits execution of a command inserted in a time period following the one just executed, without needing to wait for basic time period, defined for scheduler type, to terminate.

Parameter	Description
None	None

Result Boolean

Example:

```
Dim myObj As SchedulerCmdTarget
Public Sub Click()
        Set myObj = GetScheduler("Schedulatore")
```

```
        myObj.Reset
End Sub
```

SaveRetentive, SchedulerCmdTarget Function

Syntax SaveRetentive()

Description This function allows you to save the daily plans (normal or holidays) and the holiday dates on external files to be retained for further use after an application re-start. The file saved is the same one which can be saved with the "Holidays Scheduler" object for the "Daily plan" and "Date" schedulers. This function is managed only for "Daily plan" and "Date" schedulers, being those used for managing holidays.
The file is saved in the project's "DATA" sub folder in the following format:

<ProjectName>_<SchedulerName>.shp

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
        Dim SchedObj As SchedulerCmdTarget
        Set SchedObj = GetScheduler("Sched1")
        If Not SchedObj Is Nothing Then
                SchedObj.SaveRetentive
        End If
        Set SchedObj = Nothing
End Sub
```

CommandList, SchedulerCmdTarget Property

Syntax CommandList = _String

Description This property returns the project's XML string containing the definition of the commands associated the reference Scheduler object when the associated condition is active.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
```

```
        Dim SchedObj As SchedulerCmdTarget
        Set SchedObj = GetScheduler("Sched1")
        If Not SchedObj Is Nothing Then
                MsgBox SchedObj.CommandList,vbOkOnly,""
        End If
        Set SchedObj = Nothing
End Sub
```

CommandListOff, SchedulerCmdTarget Property

Syntax CommandListOff = _String

Description This property returns the project's XML string containing the definition of the commands associated to the reference scheduler object when the condition associated is not active.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim SchedObj As SchedulerCmdTarget
        Set SchedObj = GetScheduler("Sched1")
        If Not SchedObj Is Nothing Then
                MsgBox SchedObj.CommandListOff,vbOkOnly,""
        End If
        Set SchedObj = Nothing
End Sub
```

Enabled, SchedulerCmdTarget Property

Syntax Enabled = _Boolean

Description This property enables or disables the reference scheduler object. When the value is kept at False, the scheduler operations will not be executed.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
        Dim SchedObj As SchedulerCmdTarget
        Set SchedObj = GetScheduler("Sched1")
```

```
        If Not SchedObj Is Nothing Then
                MsgBox SchedObj.Enabled ,vbOkOnly,""
        End If
        Set SchedObj = Nothing
End Sub
```

EnableVariable, SchedulerCmdTarget Property

Syntax EnableVariable = _String

Description This property sets or returns the name of the enable variabloe for the reference scheduler object. When this property contains an empty string, Movicon will consider the 'Enable' property for enabling the object, otherwise it will consider the object enabled if the variable identified in this property obtains a value other than zero.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim SchedObj As SchedulerCmdTarget
        Set SchedObj = GetScheduler("Sched1")
        If Not SchedObj Is Nothing Then
                MsgBox SchedObj.EnableVariable,vbOkOnly,""
        End If
        Set SchedObj = Nothing
End Sub
```

HasHolidays, SchedulerCmdTarget Property

Syntax HasHolidays = _Boolean

Description This property lets you know whether the scheduler object has been set for managing holidays. This property can also be modified in runtime. When a scheduler object is not set to manage holidays, all the inherent methods ("AddHoliday", "RemoveHoliday", ecc.) will not have effect.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
```

```
        Dim SchedObj As SchedulerCmdTarget
        Set SchedObj = GetScheduler("Sched1")
        If Not SchedObj Is Nothing Then
                MsgBox SchedObj.HasHolidays,vbOkOnly, GetProjectTitle
        End If
        Set SchedObj = Nothing
End Sub
```

HolidaysPlan, SchedulerCmdTarget Property

Syntax HolidaysPlan = _Variant

Description This property allows you to set or read the current weekly hour plan of a "Daily plan" scheduler. The hourly planning managed by this function is for the holidays, whereas the normal plan is managed with the "Plan" method.

Caution. When the Scheduler is displayed through the "Scheduler Window", after having modified the planning file, you will need to use the "Scheduler Window" object's "Cancel" method to refresh the data displayed.

Parameter	Description
None	None

Result Variant

Example 1:

```
Option Explicit
Public Sub Click()
        Dim objScheduler As SchedulerCmdTarget
        Dim bHours() As Byte

        If objScheduler Is Nothing Then Set objScheduler = GetScheduler("Scheduler")

        Erase bHours
        For i = 0 To UBound(objScheduler.HolidaysPlan)
                ReDim Preserve bHours(i)
                SetVariableValue "Plan:Byte" & CStr(i), 255
                bHours(i) = 255
        Next
        objScheduler.HolidaysPlan = CVar(bHours)
End Sub
```

Example 2:

```
Option Explicit
Public Sub Click()
        Dim objScheduler As SchedulerCmdTarget
        Dim sPlan As String

        If objScheduler Is Nothing Then Set objScheduler = GetScheduler("Scheduler")

        For i = 0 To UBound(objScheduler.HolidaysPlan)
                If sPlan <> "" Then sPlan = sPlan & ", "
                sPlan = sPlan & objScheduler.HolidaysPlan(i)
        Next
        MsgBox sPlan, vbOkOnly, "Plan"
End Sub
```

Name, SchedulerCmdTarget Property

Syntax Name = _String

Description This function returns a string with the name of the reference scheduler object.

Parameter	Description
None	None

Result String

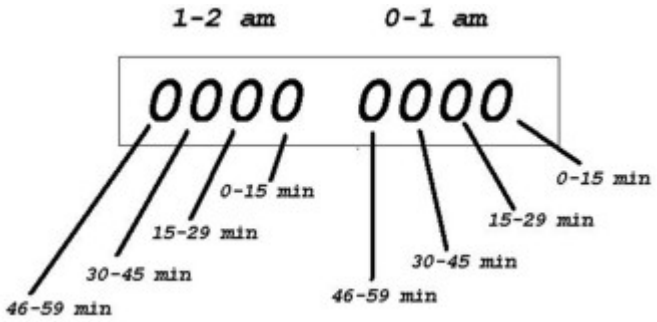
Example:

```
Option Explicit
Public Sub Click()
        Dim SchedObj As SchedulerCmdTarget
        Set SchedObj = GetScheduler("Sched1")
        If Not SchedObj Is Nothing Then
                MsgBox SchedObj.Name ,vbOkOnly,""
        End If
        Set SchedObj = Nothing
End Sub
```

Plan, SchedulerCmdTarget Property

Syntax Plan = _Variant

Description This property returns a 84 byte array (0-83). This structure provides 1 byte for each 2 hours of planning, starting from Sunday where the first byte contains information from Sunday 0-2 am, byte 1 from Sunday 2-4 and so forth. Therefore there are 12 bytes for each day of the week. However only 4 bits are dedicated to each hour of programming.
When this property is written all the byte arrays need to be set first in the this property as shown below.
The coding would therefore result as:



The first byte codes the hour from midnight to Sunday two am.
The 4 less significant bits code the time from midnight to 1 am while the most significant code the time from 1 to 2 am. The first 15 minutes are coded by starting with the 0 bit and the rest follow suit.

Attention. When the Scheduler is displayed through the "Scheduler Window" object you will need to use the Scheduler Window's "Cancel" method after editing its plan to refresh the data displayed.

Parameter	Description
None	None

Result Variant

Example:

```
Option Explicit
Public Sub Click()
        Dim objScheduler As SchedulerCmdTarget
        Dim i As Integer
        Dim sPlan As String
        Dim arrPlan(83) As Byte

        Set objScheduler = GetScheduler("Schedulatore")
        'Read actual Plan
        sPlan = ""
        For i = 0 To UBound(objScheduler.Plan)
                sPlan = sPlan & " " & CStr(objScheduler.Plan(i))
                arrPlan(i) = objScheduler.Plan(i)
        Next i

        Debug.Print "Old Plan = " & sPlan

        'Set new Plan
        sPlan = ""
        For i = 0 To 83
                arrPlan(i) = 51
                sPlan = sPlan & " " & CStr(arrPlan(i))
        Next i

        'Set Plan property
        objScheduler.Plan() = arrPlan
        Debug.Print "New Plan = " & sPlan
        Set objScheduler = Nothing
End Sub
```

TimeAndDate, SchedulerCmdTarget Property

Syntax TimeAndDate = _Date

Description This property sets or returns the command's activation time when a day or month has been selected in the "Type" property. If the "Fixed Date" has been selected instead, this property will set or return the date of the command's activation.

Parameter	Description
None	None

Result Date

Example:

```
Option Explicit
Public Sub Click()
```

```
        Dim objScheduler As SchedulerCmdTarget

        Set objScheduler = GetScheduler("Scheduler1")
        If Not objScheduler Is Nothing Then
                MsgBox objScheduler.TimeAndDate,vbInformation,GetProjectTitle
        End If
        Set objScheduler = Nothing
End Sub
```

Type, SchedulerCmdTarget Property

Syntax Type = _Integer

Description This property sets or returns the scheduler "Type".

The possible settings are:

enum_ST_DATE
enum_ST_DAY
enum_ST_FRIDAY
enum_ST_HOUR
enum_ST_MINUTE
enum_ST_MONDAY
enum_ST_MONTH
enum_ST_PLAN
enum_ST_SATURDAY
enum_ST_SUNDAY
enum_ST_THURSDAY
enum_ST_TUESDAY
enum_ST_WEDNESDAY

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
        Dim objScheduler As SchedulerCmdTarget

        Set objScheduler = GetScheduler("Scheduler1")
        If Not objScheduler Is Nothing Then
                objScheduler.Type =  enum_ST_DATE
                objScheduler.TimeAndDate  = CDate("11/04/2006 11:00:00")
        End If
        Set objScheduler = Nothing
End Sub
```

TreatHolidaysAsSunday, SchedulerCmdTarget Property

Syntax TreatHolidaysAsSunday = _Boolean

Description This property lets you know if the scheduler object has been set to manage holidays with Sunday in a normal hour plan. This property can be modified in runtime. This property is only considered when the scheduler has the Holidays option active as well.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
        Public Sub Click()
        Dim SchedObj As SchedulerCmdTarget
        Set SchedObj = GetScheduler("Sched1")
        If Not SchedObj Is Nothing Then
        MsgBox SchedObj.TreatHolidaysAsSunday, vbOkOnly, GetProjectTitle
        End If
        Set SchedObj = Nothing
End Sub
```

51. ScriptMEInterface

Loading, ScriptMEInterface Event

Description Event notified when the Basic Script resource is loaded in memory and executed the first time. After the first time, the 'Loading' routine will no longer be executed unless a "Unload" Basic Scrip command is evoked.

Parameter	Description
None	None

Main, ScriptMEInterface Event

Description Event which is executed every time the Basic Script resource is put into run mode by a command from the project. The first time the Script is called the "Loading" routine will be executed first following by the "Main" routine. Afterwards only the "Main" routine will be executed unless the Basic Script "Unload" command is evoked.

Parameter	Description
None	None

Unloading, ScriptMEInterface Event

Description Event notified when the Basic Script resource is unloaded from memory upon "Unload" command or project abort.

Parameter	Description
None	None

EnterGlobalCriticalSection, ScriptMEInterface Function

Syntax EnterGlobalCriticalSection(_nTimeout)

Description This instruction enables entry into critical processing mode for synchronizing basic scripts in separate threads. This means that the codes following the instructions will be given priority and will be synchronized, blocking any other basic scripts running the same instruction, until the

"LeaveCriticalSection' has been reached. Timeout, in milliseconds, is equal to the value passed with nTimeout.

Parameter	Description
nTimeout As Long	Timeout value.

Result Boolean

Example:

```
Sub Main
        Dim vResult As Long
        vResult = This.EnterGlobalCriticalSection(5000)
        MsgBox "EnterGlobalCriticalSection = " & CBool(vResult),vbInformation,GetProjectTitle
        vResult = This.LeaveGlobalCriticalSection
        MsgBox "LeaveGlobalCriticalSection = " & CBool(vResult),vbInformation,GetProjectTitle
End Sub
```

GetCurrentUser, ScriptMEInterface Function

Syntax GetCurrentUser()

Description Gets the User currently logged on.

Parameter	Description
None	None

Result Object
If Function has been executed successfully it will retrieve an object of type UserCmdTarget if otherwise Nothing is returned.

Example:

```
Sub Main
        Dim vResult As UserCmdTarget
        Set vResult = This.GetCurrentUser
        'do something
        Set vResult = Nothing
End Sub
```

GetInstanceNumber, ScriptMEInterface Function

Syntax GetInstanceNumber()

Description This function gets the instance number relating to the reference basic script.

Parameter	Description
None	None

Result Long

Example:

```
Sub Main
        Dim vResult As Long
        vResult = This.GetInstanceNumber()
        MsgBox "GetInstanceNumber = " & vResult,vbInformation,GetProjectTitle
End Sub
```

GetParameter, ScriptMEInterface Function

Syntax GetParameter(_nParam)

Description This function gets the index parameter equal to nParam with which the basic script was called.

Parameter	Description
nParam As Integer	parameter index.

Result String

Example:

```
Sub Main
        Dim vResult As String
        vResult = This.GetParameter(0)
        MsgBox "Parameter0 = " & vResult,vbInformation,GetProjectTitle
End Sub
```

HasPreviousInstance, ScriptMEInterface Function

Syntax HasPreviousInstance()

Description This function gets information on whether the basic script has already been instantiated.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Main
        Dim vResult As Boolean
        vResult = This.HasPreviousInstance
        MsgBox "HasPreviousInstance = " & vResult,vbInformation,GetProjectTitle
End Sub
```

IsStopping, ScriptMEInterface Function

Syntax IsStopping()

Description This function returns true when someone is trying to stop the basic script.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Main
        Dim vResult As Boolean
        Do
        vResult = This.IsStopping
        If vResult Then
        '...
        'Do something
        MsgBox "IsStopping = " & vResult,vbInformation,GetProjectTitle
        '...
        End If
        DoEvents
        Loop Until IsInStoppingMode Or vResult
End Sub
```

LeaveGlobalCriticalSection, ScriptMEInterface Function

Syntax LeaveGlobalCriticalSection()

Description This instruction leaves the critical processing mode for the basic multithread synchronization. This means that the code following the instruction will be considered as normal and no longer as priority. This instruction cancels the "EnterCriticalSection" instruction and leaves the execution to the first pending Basic Script in the "EnterCriticalSection".

Parameter	Description

None	None

Result Boolean

Example:

```
Sub Main
        Dim vResult As Long
        vResult = This.EnterGlobalCriticalSection(5000)
        MsgBox "EnterGlobalCriticalSection = " & CBool(vResult),vbInformation,GetProjectTitle
        vResult = This.LeaveGlobalCriticalSection
        MsgBox "LeaveGlobalCriticalSection = " & CBool(vResult),vbInformation,GetProjectTitle
End Sub
```

RunningOnServerSide, ScriptMEInterface Function

Syntax RunningOnServerSide()

Description This returns true when the script in running on the server project.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Main
        Dim vResult As Boolean
        vResult = This.RunningOnServerSide()
        MsgBox "RunningOnServerSide = " & vResult,vbInformation,GetProjectTitle
End Sub
```

ShowDebuggerWnd, ScriptMEInterface Function

Syntax ShowDebuggerWnd()

Description This instruction allows you to display the basic script's debug window.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Main
        Dim vResult As Boolean
        vResult = This.ShowDebuggerWnd
        MsgBox "ShowDebuggerWnd = " & vResult,vbInformation,GetProjectTitle
End Sub
```

MaxInstances, ScriptMEInterface Property

Syntax MaxInstances = _Long

Description This property sets or returns the value relating to the maximum number of contemporary instances of a basic script.

Parameter	Description
lpar As Long	Maximum number of instances.

Result Long

Example:

```
Sub Main
        MsgBox "MaxInstances = " & This.MaxInstances , vbOkOnly, GetProjectTitle
End Sub
```

ModalDialog, ScriptMEInterface Property

Syntax ModalDialog = _Boolean

Description This property sets or returns the enabling value for the 'modal dialog windows' property. When set at true the dialog windows and message boxes called within the script will be turned into modal type.

Parameter	Description
lpar As Boolean	Enabling of dialog windows to display as modal types.

Result Boolean

Example:

```
Sub Main
        MsgBox "ModalDialog = " & This.ModalDialog, vbOkOnly, GetProjectTitle
End Sub
```

RunAtServer, ScriptMEInterface Property

Syntax RunAtServer = _Boolean

Description This property sets or returns the value relating to the basic script's 'Run at Server' property. The set value turns to true when the script is run at server.

Parameter	Description
lpar As Boolean	Run at Server Enabling.

Result Boolean

Example:

```
Sub Main
        MsgBox "RunAtServer = " & This.RunAtServer, vbOkOnly, GetProjectTitle
End Sub
```

SeparateThread, ScriptMEInterface Property

Syntax SeparateThread = _Boolean

Description This property sets or returns the value relating to the basic script's 'Run in Separate Thread' property. The set value returns true when the script is run in a separate threads.

Parameter	Description
lpar As Boolean	Run in separate thread enabling.

Result Boolean

Example:

```
Sub Main
        MsgBox "SeparateThread = " & This.SeparateThread, vbOkOnly, GetProjectTitle
End Sub
```

SleepExecution, ScriptMEInterface Property

Syntax SleepExecution = _Long

Description This property sets or returns the value (in milliseconds) relating to the basic script's sleep property. The longer you set the sleep time for, the lesser the basic script will occupy the processor which will cause a slow down in its execution.

Parameter	Description

lpar As Long	Tempo di sleep.

Result Long

Example:
```
Sub Main
        MsgBox "SleepExecution = " & This.SleepExecution, vbOkOnly, GetProjectTitle
End Sub
```

StatusVariable, ScriptMEInterface Property

Syntax StatusVariable = _String

Description This property sets or returns the name of the variable relating to the basic script's status variable property.

Parameter	Description
lpar As String	Status variable.

Result Long

Example:
```
Sub Main
        MsgBox "StatusVariable = " & This.StatusVariable , vbOkOnly, GetProjectTitle
End Sub
```

SyncroScriptTimeout, ScriptMEInterface Property

Syntax SyncroScriptTimeout = _Long

Description This property sets or returns the value (in milliseconds) relating to the basic script's SyncroTimeout property. The set value refers to the milliseconds needed before stopping a syncro script.

Parameter	Description
lpar As Long	Timeout value.

Result Long

Example:
```
Sub Main
        MsgBox "SyncroScriptTimeout = " & This.SyncroScriptTimeout, vbOkOnly, GetProjectTitle
End Sub
```

ThreadPriority, ScriptMEInterface Property

Syntax ThreadPriority = _Byte

Description This property sets or returns the value relating to the basic script's execution priority property.

This property can be set with the following values:

0 Below normal
1 Normal
2 Above normal

Parameter	Description
lpar As Byte	Priority value.

Result Byte

Example:

```
Sub Main
        MsgBox "ThreadPriority = " & This.ThreadPriority, vbOkOnly, GetProjectTitle
End Sub
```

UseOwnTrace, ScriptMEInterface Property

Syntax UseOwnTrace = _Boolean

Description This property allows debug messages to be recorded on an appropriate basic script log table.

Parameter	Description
lpar As Boolean	Record debug messages enabling.

Result Boolean

Example:

```
Sub Main
        MsgBox "UseOwnTrace = " & This.UseOwnTrace, vbOkOnly, GetProjectTitle
End Sub
```

UseUIInterface, ScriptMEInterface Property

Syntax UseUIInterface = _Boolean

Description This property sets or returns the value relating to the basic script's 'UI Interface' property. The set value returns true when the UI interface has been inserted in the script. When the IU interface is inserted you will be able to access the basic script's methods and properties described in the chapter on UIInterface.

Parameter	Description
lpar As Boolean	UI Interface enabling.

Result Boolean

Example:

```
Sub Main
        MsgBox "UseUIInterface = " & This.UseUIInterface, vbOkOnly, GetProjectTitle
End Sub
```

52. SynopticCmdTarget

Click, Generic Event

Description Event occurs when the left or right mouse button is pressed within the design area.

Parameter	Description
None	None

DblClick, Generic Event

Description Event occurs when the right mouse key is double clicked within the design area. The double clicking time is set in operating system's settings.

Parameter	Description
None	None

KeyDown, Generic Event

Description Event occurs when a key is pressed down on the keyboard. This event returns the integer, KeyCode and Shift variables. This event is generated independently from being focused on.

Parameter	Description
KeyCode As Integer	Pressed Keys VBA Code. The VBA code is a set of constants which, in addition to the normal alphanumeric characters without lower/Uppercase distinction, also contemplates other keyboard keys such as the function keys, Caps Lock, etc.
Shift As Integer	Indices whether the Shift, Ctrl and Alt keys are pressed: 1 = SHIFT 2 = CTRL 4 = ALT

KeyPress, Generic Event

Description Event occurs when a key from the keyboards is pressed and released. This event returns the KeyAscii integer variable containing the pressed key's ASCII code. This event is generated only when the design is focused on.

Parameter	Description
Keyascii As Integ	The pressed key's ASCII code.

KeyUp, Generic Event

Description Event occurs when a key on the keyboard is released (after being pressed). This event releases the interger type keyCode and Shift variables. This event occurs indipendently of being focused on.

Parameter	Description
KeyCode As Integer	The pressed key's VBA code. The VBA code is a set of constants that, apart from the normal alphanumeric characters, without upper/lowercase distinction, contemplates other keys such as the Caps Lock function key etc.
Shift As Integer	Indicates whether whether the Shift, Ctrl and Alt keys are pressed: 1 = SHIFT 2 = CTRL 4 = ALT

KillFocus, Generic Event

Description Event occurs when the object in question is deselected or loses focus.

Parameter	Description
None	None

MouseDown, Generic Event

Description Event notified both in the screen code and in the object code every time the mouse key is clicked on screen, independently from its position or symbol. This event returns the integer Button and Shift type variables and the X and Y single type variables.
In order to manage this event only within a screen object you will need to use the "IsCursorOnObject" function.

Parameter	Description
Button As Integer	Indicates pressed mouse button: 1 = Left 2 = Right 4 = Central
Shift As Integer	Indicates whether the Shift, Ctrl and Alt keys are pressed: 1 = SHIFT 2 = CTRL 4 = ALT

X As Single	Horizontal coordinates referring to the cursor's position when event occurs.
Y As Single	Vertical coordinates referring to the cursor's position when event occurs.

MouseMove, Generic Event

Description Event notified both in the screen code and the object code when the mouse cursor changes position on screen, independently from the position or symbol. This event returns the Button and Shift integer type variables and the X and Y single type variables.
In order to manage this evenly only within a screen object you will need to use the "IsCursorOnObject" function.

Parameter	Description
Button As Integer	Pressed mouse key index: 1 = Left 2 = Right 4 = Central
Shift As Integer	Indicates whether the Shift, Ctrl and Alt keys are pressed: 1 = SHIFT 2 = CTRL 4 = ALT
X As Single	Horizontal coordinate referring to the cursor's positon when event occurs.
Y As Single	Vertical coordinate referring to the cursor's position when event occurs.

MouseUp, Generic Event

Description Event notified both in the screen and object codes when any one of the mouse keys are released on screen, independently from its position or symbol. This event returns the Button and Shift integer type variables and the X and Y single type variables.
In order to manage this event only within an object on screen you will need to use the "IsCursorOnObject" function.

Parameter	Description
Button As Integer	Pressed mouse key index: 1 = Left 2 = Right 4 = Central
Shift As Integer	Indicates whether the Shft, Ctrl and Alt keys are pressed: 1 = SHIFT 2 = CTRL 4 = ALT
X As Single	Horizontal coordinates referring to the cursor's positon when event occurs
Y As Single	Vertical coordinates referring to the cursor's position when event occurs

OnActivate, SynopticCmdTarget Event

Description Event generated when the screen is made active or deactive. This event returns a boolean parameter indicating the activation status ,which means whether it's being focused on or not.

Parameter	Description
bActive As Boolean	Activation status: True = Screen has been activated False = Screen has been deactivated

OnQueryEndSession, SynopticCmdTarget Event

Description Event generated following a close screen command. The screen's closure can be blocked by setting it at false by means of using the bRet parameter.

Parameter	Description
bRet As Boolean	Enabling closure: True = consents to closing the screen False = blocks the screen from being closed.

OnSize, SynopticCmdTarget Event

Description This event verifies when the screen window resizes (not when minimized).

Parameter	Description
nWidth As Integer	Out parameter. returns screen length.
nHeight As Integer	Out parameter. Returns screen height.

Example:

```
Public Sub OnSize(ByRef nWidth As Integer, ByRef nHeight As Integer)
        MsgBox "Screen width = " & nWidth & ", Screen height =  " & nHeight
End Sub
```

OnStartSynapsisExecution, SynopticCmdTarget Event

Description Event generated at the start of synapses type logic execution on screen.

Parameter	Description
None	None

OnStopSynapsisExecution, SynopticCmdTarget Event

Description Event generated at the end of synapses type logic execution on screen.

Parameter	Description
None	None

SetFocus, Generic Event

Description Event occurs when the design object receives focus or is selected.

Parameter	Description
None	None

SynopticLoading, SynopticCmdTarget Event

Description Event notified when the Screen is loaded in memory.

Parameter	Description
None	None

SynopticUnloading, SynopticCmdTarget Event

Description Event notified when the Screen is unloaded from memory.

Parameter	Description
None	None

CloseSynoptic, SynopticCmdTarget Function

Syntax CloseSynoptic()

Description This function closes the screen. This function has not effect when called from a startup screen. It is used to close screens opened in modal mode or in a separate frame. Once this function is called you will be returned back to the startup screen.

Parameter	Description
None	None

Result None

Example:
Public Sub Click()
CloseSynoptic()
End Sub

CreateNewSymbol, SynopticCmdTarget Function

Syntax CreateNewSymbol(_lpszSymbolName, _lpszSymbolCode, nSymbolType)

Description This function allows a new symbol to be created on screen. Accepts two string parameters containing the name in the symbol and the basic code within the same symbol respectively and a integer parameter indicating the symbol type. All the new symbol's features can be configured through the basic properties. Although there are no capacity restrictions for string variables, it may be handy to write to an external text file the whole script code.

0 = rectangle
1 = rounded rectangle
2 = arc
3 = polybezier
4 = pie
5 = Ellipse
6 = line
7 = text
8 = embedded screen
9 = trend
10 = Gauge

This property is only partly supported in Windows CE. (provided in creating arc, chord and pie symbols)

Parameter	Description
lpszSymbolName As String	Object name.
lpszSymbolCode As String	Object script code.
nSymbolType As Integer	Symbol type.

Result Object
If Function has been executed successfully it will retrieve an object of type DrawCmdTarget if otherwise Nothing is returned.

Example:

```
Public Sub Click()
        Dim lpszSymbolName As String
        Dim lpszSymbolCode As String
        Dim nSymbolType As Integer
        Dim sType As String
        Dim obj As Object
        Dim bErr As Boolean
        ' Ask the type (Chide il tipo)
        sType = InputBox("Symbol's type:", "CreateNewSymbol", "0", 100, 100)
        If sType <> "" And IsNumeric(sType) Then
                lpszSymbolName = "MySymbol"
                nSymbolType = CInt(sType)
                lpszSymbolCode = "'#Uses ""Codice.txt"""
                ' Destroy the symbol (Distrugge il simbolo)
                DestroySymbol(lpszSymbolName)
                ' Create the symbol (Crea il simbolo)
                bErr = False
                On Error GoTo NoObj
                Set obj = CreateNewSymbol(lpszSymbolName, "", nSymbolType)
                 ' Show the symbol (Visualizza il simbolo)
                 obj.Height = 100
                 obj.Width = 100
                 obj.Xpos = 500
                 obj.Ypos = 300
                 obj.BackColor = RGB(192,192,192)
                 On Error GoTo 0
                 Set obj = Nothing
        End If
        Exit Sub
        NoObj:
        If Not bErr Then
                 Debug.Print "Object is Nothing: " & nSymbolType
                 bErr = True
        End If
        Resume Next
End Sub
```

DestroySymbol, SynopticCmdTarget Function

Syntax DestroySymbol(_lpszSymbolName)

Description This function allows the symbol to be deleted from the screen. Accepts a string parameter containing the name of the symbol to be deleted. This function can work on symbols inserted during the programming mode and on symbols previously created by means of the CreateNewSymbol function. The function returns a boolean value either in True when deletion is successful or in False when not. One reason for failing may be due to deleting a non-existent symbol.

Parameter	Description
lpszSymbolName As String	Name of symbol.

Result Boolean

Example:

```
Public Sub Click()
        Dim lpszSymbolName As String
        Dim lpszSymbolCode As String
        Dim nSymbolType As Integer
        Dim sType As String
        Dim obj As Object
        Dim bErr As Boolean
        ' Ask the type (Chide il tipo)
        sType = InputBox("Symbol's type:", "CreateNewSymbol", "0", 100, 100)
        If sType <> "" And IsNumeric(sType) Then
                lpszSymbolName = "MySymbol"
                nSymbolType = CInt(sType)
                lpszSymbolCode = "'#Uses ""Codice.txt"""
                ' Destroy the symbol (Distrugge il simbolo)
                DestroySymbol(lpszSymbolName)
                ' Create the symbol (Crea il simbolo)
                bErr = False
                On Error GoTo NoObj
                Set obj = CreateNewSymbol(lpszSymbolName, "", nSymbolType)
                 ' Show the symbol (Visualizza il simbolo)
                 obj.Height = 100
                 obj.Width = 100
                 obj.Xpos = 500
                 obj.Ypos = 300
                 obj.BackColor = RGB(192,192,192)
                 On Error GoTo 0
                 Set obj = Nothing
        End If
        'Wait 1 second (aspetta un secondo)
        Wait 5
        DestroySymbol(lpszSymbolName)
        Exit Sub
        NoObj:
        If Not bErr Then
                 Debug.Print "Object is Nothing: " & nSymbolType
                 bErr = True
        End If
        Resume Next
End Sub
```

GetAbsoluteSubObject, SynopticCmdTarget Function

Syntax GetAbsoluteSubObject(_lpszName)

Description This function permits you to access the on screen object's methods and properties even when it is contained in a symbol. The lpszName parameter identifies the name of the object.

This function is not supported in Windows CE.(if used, always returns 'null')

Parameter	Description
lpszName As String	Name of object.

Result Object
If Function has been executed successfully it will retrieve an object of type DrawCmdTarget if otherwise Nothing is returned.

Example:

```
Public Sub Click()
        Dim obj As Object
        ' Create object (Crea oggetto)
        Set obj = GetAbsoluteSubObject("Object1")
        obj.BackColor = RGB(192,192,192)
        Set obj = Nothing
End Sub
```

GetActiveUserObject, SynopticCmdTarget Function

Syntax GetActiveUserObject()

Description This function allows you to retrieve the active user object for the screen. By using this method you can find out which Web Client user is logged on to the Server. When there are no users logged on, this function will return a Nothing object.
The same named 'UserAndGroupCmdTarget' basic interface method cannot be used for the previously described purpose. This method when used for a Web Client returns the last user who logged on to the Server and not the one actually logged on the Web Client.

Parameter	Description
None	None

Result Object
If Function has been executed successfully it will retrieve an object of type UserCmdTarget if otherwise Nothing is returned.

Example:

```
Option Explicit
Public Sub Click()
        Dim objUser As UserCmdTarget

        Set objUser = GetSynopticObject.GetActiveUserObject
        If Not objUser Is Nothing Then
                MsgBox(objUser.Name & " is Logged In", vbOkOnly, GetProjectTitle)
                Set objUser = Nothing
        End If
End Sub
```

GetAlias, SynopticCmdTarget Function

Syntax GetAlias(_lpszAlias)

Description This function returns the value defined for the Alias passes as "lpszAlias" parameter defined in the screen's Alias Table.

Parameter	Description
lpszAlias As String	Name of Alias where value is to be retrieved.

Result String

Example:

```
Public Sub Click()
        Dim objScreen As SynopticCmdTarget
        Set objScreen = GetSynopticObject
        MsgBox "Alias <<TsetAlais>> = "
        objScreen.GetAlias("TsetAlais"),vbInformation, GetProjectTitle
        Set objScreen = Nothing
End Sub
```

GetAliasListName, SynopticCmdTarget Function

Syntax GetAliasListName()

Description This function returns the list of Aliases defined in the screen. This function returns the list of Aliases defined in the screen. A string will be returned where the names of the Aliases are separated by the "|" (pipe) character.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim objScreen As SynopticCmdTarget
        Set objScreen = GetSynopticObject
        MsgBox "Alias List = " & objScreen.GetAliasListName(),vbInformation,
        GetProjectTitle
        Set objScreen = Nothing
End Sub
```

GetAliasListValue, SynopticCmdTarget Function

Syntax GetAliasListValue()

Description This function returns the list of values associated to the Aliases defined in the screen. A string will be returned where the values the Aliases are separated by the "|" (pipe) character.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim objScreen As SynopticCmdTarget
        Set objScreen = GetSynopticObject
        MsgBox "Alias Value List = " & objScreen.GetAliasListValue(),vbInformation,
        GetProjectTitle
        Set objScreen = Nothing
End Sub
```

GetAppTimeZone, PmeDocCmdTarget Function

Syntax GetAppTimeZone

Description This function informs you on the timezone in minutes set in the computer where the project is being run.
The returned timezone is always considers any legal time in act.

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        Dim sMsg As String
        sMsg = CStr(GetAppTimeZone) & " min."
        MsgBox "Time Zone : " & sMsg, vbOkOnly + vbInformation, GetProjectTitle
End Sub
```

GetFocus, SynopticCmdTarget Function

Syntax GetFocus()

Description This function permits you to access methods and properties of the object which have focus on screen.
If none of the objects have focus the object variable is Nothing.

Parameter	Description

None	None

Result Object
If Function has been executed successfully it will retrieve an object of type DrawCmdTarget if otherwise Nothing is returned.

Example:

```
Public Sub MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
        Dim obj As Object

        ' Create object (Crea l'oggetto)
        Set obj = GetFocus
        ' Print title (Stampa titolo)
        On Error Resume Next
        Debug.Print obj.title
        On Error GoTo 0
        ' Delete object (Distrugge l'oggetto)
        Set obj = Nothing
End Sub
```

GetImage, SynopticCmdTarget Function

Syntax GetImage(_nImageType)

Description This function allows you to get the image stored in memory with a certain quality. The image's quality is defined in the nImage Type parameter. The result contains the image.
The nImage Type parameter can have the following values:

0 High quality
1 Medium-high quality
2 Medium-low quality
3 Low quality

This function is not fully supported in Windows CE. (the image should always return as bitmatp only)

This function loads the image only if the screen is open. Even though the screen is kept in memory because its "Keep in memory" option has been marked, when closed the image will nevertheless be unloaded from memory and the GetImage function will no longer be capable of loading it again. The GetImage function will return a variant containing the image if loaded successfully, otherwise and it will return an empty variant if not loaded successfully.

Parameter	Description
nImageType As Integer	Image quality.

Result Variant

Example:

```
Public Sub Click()
        Dim vResult As Variant
        Dim nImageType As Integer
```

```
        nImageType = 1
        vResult = GetImage(nImageType)
End Sub
```

GetInstanceNumber, SynopticCmdTarget Function

Syntax GetInstanceNumber()

Description This function returns the index assigned to the current instance with which the screen has been opened. Each instance presents a different index number.

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        Dim objScreen As SynopticCmdTarget

        Set objScreen = GetSynopticObject
        Debug.Print objScreen.GetInstanceNumber()
        Set objScreen = Nothing
End Sub
```

GetObjectByUniqueID, SynopticCmdTarget Function

Syntax GetObjectByUniqueID(_ID)

Description This function allows you to access the method and properties of the object on the screen. The object must contain synapses in order to be referenced with ID parameter. Movicon automatically assigns a unique ID number each object containing synapses and the ID parameter is used for indicating the object you wish to access. The GetUniqueObjectID function from the SymbolInterface is used for returning drawing ID numbers.

Parameter	Description
ID As Long	Object ID.

Result Object
If Function has been executed successfully it will retrieve an object of type DrawCmdTarget if otherwise Nothing is returned.

Example:

```
Public Sub OnStartSynapsisExecution()
        Dim obj as DrawCmdTarget
```

```
        'Create object (Crea oggetto)
        Set obj = GetObjectByUniqueID(3)
        ' Changhe color (Cambia colore)
        On Error GoTo NoObj
        obj.BackColor = RGB(192,192,192)
        On Error GoTo 0
        Set obj = Nothing
        Exit Sub

        NoObj:
        Debug.Print "Not object ID: 3"
        Resume Next
End Sub
```

GetParameterVariable, SynopticCmdTarget Function

Syntax GetParameterVariable(_lpszVariableName)

Description This function returns the name of the variable which replaced the variable referenced with the lspzVariableName parameter in the screen's parameter file. When a screen is parameter driven this means that the variables used in symbols execution properties can be replaced with others specified in the object's text file. Please refer to the ParameterFileName property for the Parameter File settings.

Parameter	Description
lpszVariableName As String	Variable's name.

Result String

Example:

```
Public Sub Click()
        Dim lpszVariableName As String
        Dim sResult As String

        ParameterFileName = "Parametri.txt"
        lpszVariableName = InputBox("Variable name?", "", "VAR00001", 100, 100)
        sResult = GetParameterVariable(lpszVariableName)
        MsgBox "GetParameterVariable= " & sResult , vbOkOnly, GetProjectTitle
End Sub
```

GetSubObject, SynopticCmdTarget Function

Syntax GetSubObject(_lpszObjectName)

Description This function allows symbols, contained in the screen, to be referenced. Accepts a string parameter containing the name of the object to be given a reference. Please remember that when dealing with simple objects, the contents from the object's "Name" property must be passed and when dealing with composed objects the contents from the object's "Description" property must be passed.

Parameter	Description

lpszObjectName As String	Object name.

Result Object
If Function has been executed successfully it will retrieve an object of type DrawCmdTarget if otherwise Nothing is returned.

Example:

```
Public Sub Click()
        Dim vResult As DrawCmdTarget
        Set vResult = GetSubObject("Object1")
        vResult .BackColor = vResult.BackColor + 10
        Set vResult = Nothing
End Sub
```

GetTimeZone, SynopticCmdTarget Function

Syntax GetTimeZone()

Description This function informs you on the time sine in minutes set tin the computer where the screen is displayed. In cases where the screen is displayed by a Web Client station the value will indicated the time zone of the computer in which the browser is used for magaging the html page.
The returned time zone always considers any legal time in act.

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        Dim sMsg As String
        sMsg = CStr(GetSynopticObject.GetTimeZone) & " min."
        MsgBox "Time Zone : " & sMsg, vbOkOnly + vbInformation, GetProjectTitle
End Sub
```

GetWindowPos, SynopticCmdTarget Function

Syntax GetWindowPos(_nX, _nY, _nWidth, _nHeight, _nShow)

Description This function is read only and permits information about the referenced screen size and position to be read.

This function is only partly supported in Windows CE.(nShow can only obtain the '0'(SW_HIDE)o '5'(SW_SHOW) values)

SW_HIDE
Hides the window and activates another window.
SW_MAXIMIZE
Maximizes the specified window.
SW_MINIMIZE
Minimizes the specified window and activates the next top-level window in

the z-order.
SW_RESTORE
Activates and displays the window. If the window is minimized or maximized, the system restores it to its original size and position. An application should specify this flag when restoring a minimized window.
SW_SHOW
Activates the window and displays it in its current size and position.
SW_SHOWMAXIMIZED
Activates the window and displays it as a maximized window.
SW_SHOWMINIMIZED
Activates the window and displays it as a minimized window.
SW_SHOWMINNOACTIVE
Displays the window as a minimized window.
This value is similar to SW_SHOWMINIMIZED, except the window is not activated.
SW_SHOWNA
Displays the window in its current size and position.
This value is similar to SW_SHOW, except the window is not activated.
SW_SHOWNOACTIVATE
Displays a window in its most recent size and position.
This value is similar to SW_SHOWNORMAL, except the window is not actived.
SW_SHOWNORMAL
Activates and displays a window. If the window is minimized or maximized, the system restores it to its original size and position. An application should specify this flag when displaying the window for the first time.

Parameter	Description
nX As Integer	top left corner X coordinate.
nY As Integer	top right corner Y coordinate.
nWidth As Integer	width size.
nHeight As Integer	height size.
nShow As Integer	screen display mode

Result Boolean

Example:

```
Public Sub Click()
        Dim nX As Variant
        Dim nY As Variant
        Dim nWidth As Variant
        Dim nHeight As Variant
        Dim nShow As Variant
        GetWindowPos(nX , nY , nWidth , nHeight , nShow )
        MsgBox "nX = " & nX  &" nY = " & nY  & "nWidth = " & nWidth & "nHeight = " & nHeight &
        "nShow = " & nShow , vbOkOnly, GetProjectTitle
End Sub
```

IsRemoteClientView, SynopticCmdTarget Function

Syntax IsRemoteClientView()

Description Returns True when the basic code has been invoked by a Movicon WebClient object.

Parameter	Description
None	None

Result Boolean

Example:
Public Sub Click()
MsgBox "IsRemoteClientView = " & CStr(**IsRemoteClientView**), vbOkOnly, GetProjectTitle
End Sub

PrintSynoptic, SynopticCmdTarget Function

Syntax PrintSynoptic(_nMode, _bKeepPrintProportions)

Description Prints the screen. Accepts the nMode parameter indicating the print mode.

The parameter can have the following the values:
0=select printer
1=direct printout
2=Preview

Parameter	Description
nMode As Integer	Print Mode.
bKeepPrintProportions as boolean	Optional Parameter. When set at "True" permits prints exactly to the proportions as seen on screen. When set at 'False' (default value) both height and width are adapted to fit within screen page.

Result Boolean

Example:
Public Sub Click()
PrintSynoptic(2)
End Sub

RemoveAlias, SynopticCmdTarget Function

Syntax RemoveAlias(_lpszAlias)

Description This function removes the Alias passed as the "lpszAlias" parameter.

Parameter	Description
lpszAlias As String	Name of Alias to be removed.

Result Boolean

Example:

```
Public Sub Click()
        Dim objScreen As SynopticCmdTarget
        Set objScreen = GetSynopticObject
        Debug.Print objScreen.RemoveAlias("TsetAlais")
        Set objScreen = Nothing
End Sub
```

RemoveAllAliases, SynopticCmdTarget Function

Syntax RemoveAllAliases()

Description This function removes all the Aliases defined in the screen's Table.

Parameter	Description
None	None

Result None

Example:

```
Public Sub Click()
        Dim objScreen As SynopticCmdTarget
        Set objScreen = GetSynopticObject
        Debug.Print objScreen.RemoveAllAliases()
        Set objScreen = Nothing
End Sub
```

SaveImageToFile, SynopticCmdTarget Function

Syntax SaveImageToFile(_lpszFileName, _nImageType)

Description This function allows the displayed screen to be saved as an Image. The print quality is defined by the nImageType parameter. The values which can be used are from 0 to 3.

0 High quality
1 medium-high quality
2 medium-low quality
3 low quality

In addition to indicating quality type, the nImageType parameter also indicated the format type with which the image will be saved:

0 = imf_ImageFormatBMP
1 = imf_ImageFormatPNG
2 = imf_ImageFormatJPEG
3 = imf_ImageFormatGIF

As this function is not supported in Window CE, all images will be created in ".bmp" format.

This function loads the image only if the screen is open. Even though the screen is kept in memory because its "Keep in memory" option has been marked, when closed the image will nevertheless be unloaded from memory and the GetImage function will no longer be capable of loading it again. The GetImage function will return a variant containing the image if loaded successfully, otherwise and it will return an empty variant if not loaded successfully.

Parameter	Description
lpszFileName As String	Name of file.
nImageType As Integer	Modality with which to save image.

Result Boolean

Example:

```
Public Sub Click()
        Dim bResult As Boolean
        Dim lpszFileName As String
        Dim nImageType As Integer
        lpszFileName = "Print.jpg"
        nImageType = 1
        bResult = SaveImageToFile(lpszFileName, nImageType)
End Sub
```

SetAlias, SynopticCmdTarget Function

Syntax SetAlias(_lpszAlias, _lpszValue)

Description This function sets the value defined for the Alias passed as the "lpszAlias" parameter. The new value will be the one passed with the "lpszValue " parameter and may be a variable name or a string or numeric value.
If the Alias does not exist in the object's Table it will be added as a new one.

Parameter	Description
lpszAlias As String	Name of the Alias for which the value is set. If Alias does not exist, it will be added as a new one.
lpszValue As String	Value to set the Alias with.

Result Boolean

Example:

```
Public Sub Click()
        Dim objScreen As SynopticCmdTarget
        Set objScreen = GetSynopticObject
        Debug.Print objScreen.SetAlias("TsetAlais", "VAR00001")
        Set objScreen = Nothing
End Sub
```

SetFocusTo, SynopticCmdTarget Function

Syntax SetFocusTo(_lpszObjectName)

Description This function allows the system focus to be fixed on a certain drawing on screen. Accepts the lpszObjectName parameter containing the name of the symbol to be pointed at. The returned True value indicates that the symbol has been focused on otherwise if the drawing is not found on the screen the function will return with the False value.

Parameter	Description
lpszObjectName As String	Name of object.

Result Boolean

Example:

```
Public Sub MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
        Dim lpszObjectName As String
        Dim bResult

        ' Ask the symbol (Chiede il simbolo)
        lpszObjectName = InputBox("Wath symbol do you want set focus ?",
        "SetFocusTo", "Symbol1", 100, 100)
        ' Set focus (Imposta il fuoco)
        bResult = SetFocusTo(lpszObjectName)
        If Not bResult Then MsgBox("Symbols not found !", , "SetFocusTo")
```

```
End Sub
```

SetRedraw, SynopticCmdTarget Function

Syntax SetRedraw(_bSet)

Description This function allows the graphic management to be enabled or disabled in the screen page where it was called. The bSet parameter identifies one of the two modes.
This is very delicate to use especially when being disabled as it may cause the user to lose the possibility to interact with various objects or symbols existing on the screen page. This function is handy for disabling the viewing of long graphical operations in the project, to made active only when they have been completed.

Parameter	Description
bSet As Boolean	Enabling value.

Result None

Example:

```
Public Sub Click()
        If MsgBox("Do you want to suspend redraw on the synoptic ?", vbYesNo, "SetRedraw") =
        vbYes Then
         SetRedraw(False)
         Wait 5
        End If
        SetRedraw(True)
End Sub
```

SetSynapsisVisible, SynopticCmdTarget Function

Syntax SetSynapsisVisible(_bVisible)

Description This function permits the showing of synapses and connections (connector object) on screen to be enabled or disabled. When the bVisible parameter is set at false the synapses and connections will not be displayed. True is the value set for default at the opening of a screen.

Parameter	Description
bVisible As Boolean	Enabling value.

Result None

Example:

```
Public Sub SynopticLoading()
        ' Doesn't show synapsis
        SetSynapsisVisible(False)
```

```
End Sub
```

SetWindowPos, SynopticCmdTarget Function

Syntax SetWindowPos(_nX, _nY, _nWidth, _nHeight, _nShow)

Description Permits the window containing the screen to be moved and resized.

Parameter	Description
nX As Integer	Top left corner X coordinate.
nY As Integer	Top left corner Y coordinate.
nWidth As Integer	Width size.
nHeight As Integer	Height size.
nShow As Integer	Display mode definitions: 0= hide 1= restore 2= reduce to icon 3= maximize

Result Boolean

Example:

```
Public Sub Click()
        SetWindowPos(20, 100, 500, 400, 1)
End Sub
```

ZoomIn, SynopticCmdTarget Function

Syntax ZoomIn()

Description Increases the screen's zoom factor.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim bResult As Boolean
        bResult = ZoomOut
        MsgBox "ZoomOut = " & CStr(bResult),vbOkOnly, GetProjectTitle
```

```
End Sub
```

ZoomOut, SynopticCmdTarget Function

Syntax ZoomOut()

Description Decreases the screen's zoom factor.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim bResult As Boolean
        bResult = ZoomOut
        MsgBox "ZoomOut = " & CStr(bResult),vbOkOnly, GetProjectTitle
End Sub
```

ZoomTo, SynopticCmdTarget Function

Syntax ZoomTo()

Description Returns the screen's default screen factor.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        Dim bResult As Boolean
        bResult = ZoomTo
        MsgBox "ZoomTo = " & CStr(bResult),vbOkOnly, GetProjectTitle
End Sub
```

BackColor, SynopticCmdTarget Property

Syntax BackColor = _Long

Description This property sets or returns the screen's back color.
The passed or returned value contains the back color code (R,G,B, in each byte). You may find it easier to use the RGB function for identifying the color.

Parameter	Description
None	None

Result Long

Example:

```
'Screen environment
Public Sub Click()
        Dim lRet As Long
        BackColor = RGB(255,0,0)
        lRet = BackColor
        MsgBox "BackColor = " & CStr(lRet), vbOkOnly, "Test BackColor"
        BackColor = RGB(0,255,0)
        lRet = BackColor
        MsgBox "BackColor = " & CStr(lRet), vbOkOnly, "Test BackColor"
        BackColor = RGB(0,0,255)
        lRet = BackColor
        MsgBox "BackColor = " & CStr(lRet), vbOkOnly, "Test BackColor"
End Sub
'On screen object environment
Public Sub Click()
        Dim ObjSyn As SynopticCmdTarget
        Dim lRet As Long
        Set ObjSyn = GetSynopticObject
        ObjSyn.BackColor= RGB(255,0,0)
        lRet = ObjSyn.BackColor
        MsgBox "BackColor= " & CStr(lRet), vbOkOnly, "Test BackColor"
        ObjSyn.BackColor= RGB(0,255,0)
        lRet = ObjSyn.BackColor
        MsgBox "BackColor = " & CStr(lRet), vbOkOnly, "Test BackColor"
        ObjSyn.BackColor= RGB(0,0,255)
        lRet = ObjSyn.BackColor
        MsgBox "BackColor = " & CStr(lRet), vbOkOnly, "Test BackColor"
        Set ObjSyn = Nothing
End Sub
```

BackGroundFileBitmap, SynopticCmdTarget Property

Syntax BackGroundFileBitmap = _String

Description This property sets or returns the screen's background image.
The passed or returned value contains the fixed address for extended image.

Parameter	Description
None	None

Result String

Example:

```
'Screen environment
Public Sub Click()
        MsgBox "BackGroundFileBitmap = " & BackGroundFileBitmap , vbOkOnly, GetProjectTitle
        BackGroundFileBitmap = "Images\background.bmp" 'supposing that there is background
        image.bmp in the specified directory

End Sub

Public Sub Click()
        Dim ObjSyn As SynopticCmdTarget
        Set ObjSyn = GetSynopticObject
        MsgBox "BackGroundFileBitmap = " & ObjSyn.BackGroundFileBitmap , vbOkOnly,
        GetProjectTitle
         'supposing that there is a background image.bmp in the specified directory
        ObjSyn.BackGroundFileBitmap = "Images\background.bmp"
        Set ObjSyn = Nothing
End Sub
```

BackGroundFileBitmapTile, SynopticCmdTarget Property

Syntax BackGroundFileBitmapTile = _Boolean

Description This property sets or returns the enabling of the background bitmap file repetition like tiles to cover the entire screen surface. Accepts a Boolean value.

This property is not supported in Windows CE. (always returns 'false' if used)

Parameter	Description
None	None

Result Boolean

Example:

```
'Screen environment
Public Sub Click()
        MsgBox "BackGroundFileBitmapTile = " & BackGroundFileBitmapTile , vbOkOnly,
        GetProjectTitle
        BackGroundFileBitmapTile = False
End Sub
'On screen object environment
Public Sub Click()
        Dim ObjSyn As SynopticCmdTarget
        Set ObjSyn = GetSynopticObject
        MsgBox "BackGroundFileBitmapTile = " & ObjSyn.BackGroundFileBitmapTile , vbOkOnly,
        GetProjectTitle
        ObjSyn.BackGroundFileBitmapTile = False
        Set ObjSyn = Nothing
End Sub
```

CXBackImage, SynopticCmdTarget Property

Syntax CXBackImage = _Integer

Description This property sets or returns the width size of the screen's back image.

Parameter	Description
None	None

Result Integer

Example:

```
'Screen Environment
Public Sub Click()
        Dim lRet As Integer
        lRet = CXBackImage
        MsgBox "CXBackImage = " & CStr(lRet), vbOkOnly, "Test CXBackImage "
        CXBackImage = 100
        lRet = CXBackImage
        MsgBox "CXBackImage = " & CStr(lRet), vbOkOnly, "Test CXBackImage "
End Sub

Public Sub Click()
        Dim ObjSyn As SynopticCmdTarget
        Dim lRet As Long
        Set ObjSyn = GetSynopticObject
        lRet = ObjSyn.CXBackImage
        MsgBox "CXBackImage = " & CStr(lRet), vbOkOnly, "Test CXBackImage "
        ObjSyn.CXBackImage = 100
        lRet = ObjSyn.CXBackImage
        MsgBox "CXBackImage = " & CStr(lRet), vbOkOnly, "Test CXBackImage "
        Set ObjSyn = Nothing
End Sub
```

CYBackImage, SynopticCmdTarget Property

Syntax CYBackImage = _Integer

Description This property sets or returns the height size of the screen's back image.

Parameter	Description
None	None

Result Integer

Example:

```
'Screen Environment
Public Sub Click()
        Dim lRet As Integer
```

```
        lRet = CYBackImage
        MsgBox "CYBackImage = " & CStr(lRet), vbOkOnly, "Test CYBackImage "
        CYBackImage = 100
        lRet = CYBackImage
        MsgBox "CYBackImage = " & CStr(lRet), vbOkOnly, "Test CYBackImage "
End Sub

Public Sub Click()
        Dim ObjSyn As SynopticCmdTarget
        Dim lRet As Long
        Set ObjSyn = GetSynopticObject
        lRet = ObjSyn.CYBackImage
        MsgBox "CYBackImage = " & CStr(lRet), vbOkOnly, "Test CYBackImage "
        ObjSyn.CYBackImage = 100
        lRet = ObjSyn.CYBackImage
        MsgBox "CYBackImage = " & CStr(lRet), vbOkOnly, "Test CYBackImage "
        Set ObjSyn = Nothing
End Sub
```

CyclicExecution, SynopticCmdTarget Property

Syntax CyclicExecution = _Boolean

Description When this property is enabled you will be allowed to execute any synapse logic in cyclic mode when the screen is active.

Parameter	Description
None	None

Result Boolean

Example:

```
'Screen environment
Public Sub Click()
        Dim lRet As Integer
        lRet = CyclicExecution
        MsgBox "CyclicExecution = " & CStr(lRet), vbOkOnly, "Test CyclicExecution "
End Sub
'Onscreen object environment
Public Sub Click()
        Dim ObjSyn As SynopticCmdTarget
        Dim lRet As Long
        Set ObjSyn = GetSynopticObject
        lRet = ObjSyn.CyclicExecution
        MsgBox "CyclicExecution = " & CStr(lRet), vbOkOnly, "Test CyclicExecution "
        Set ObjSyn = Nothing
End Sub
```

EnableScrollBars, SynopticCmdTarget Property

Syntax EnableScrollBars = _Boolean

Description The scroll bars will be displayed in the window containing the screen when this property is enabled. This property is valid when the screen size is bigger than the window containing it and when the FitInWindow property is not set at True.

This property is not supported in Windows CE.(if used, always returns 'false')

Parameter	Description
None	None

Result Boolean

Example:

```
'Screen Environment
Public Sub Click()
        Dim lRet As Integer
        lRet = EnableScrollBars
        MsgBox "EnableScrollBars = " & CStr(lRet), vbOkOnly, "Test EnableScrollBars "
End Sub

Public Sub Click()
        Dim ObjSyn As SynopticCmdTarget
        Dim lRet As Long
        Set ObjSyn = GetSynopticObject
        lRet = ObjSyn.EnableScrollBars
        MsgBox "EnableScrollBars = " & CStr(lRet), vbOkOnly, "Test EnableScrollBars  "
        Set ObjSyn = Nothing
End Sub
```

FastTickCounter, SynopticCmdTarget Property

Syntax FastTickCounter = _Long

Description This property allows to read or set the number of loops per tick in the pending synapses type processing or in animation managed on polling associated to the screen. It would be advisable not to change the default value. The FastTickCounter especially concerns high priority operations.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim sRet As Long
        sRet = FastTickCounter
        MsgBox "FastTickCounter = " & sRet, vbOkOnly, GetProjectTitle
End Sub
```

FastTickFrequency, SynopticCmdTarget Property

Syntax FastTickFrequency = _Long

Description This property allows you to set or display the Tick frequency for handling animations managed on polling and, for example, processing logic of synapses associated to the screen. It would be advisable not to change the default value. The FastTickFrequency especially concerns high priority operations.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim sRet As Long
        sRet = FastTickFrequency
        MsgBox "FastTickFrequency = " & sRet, vbOkOnly, GetProjectTitle
End Sub
```

FitInWindow, SynopticCmdTarget Property

Syntax FitInWindow = _Boolean

Description When this property is enabled you will be permitted to force the screen's sizes to fit in the window containing it.

Parameter	Description
None	None

Result Boolean

Example:

```
'Screen environment
Public Sub Click()
        Dim lRet As Integer
        lRet = FitInWindow
        MsgBox "FitInWindow = " & CStr(lRet), vbOkOnly, "Test FitInWindow"
End Sub
```

```
'On screen object environment
Public Sub Click()
        Dim ObjSyn As SynopticCmdTarget
        Dim lRet As Long
        Set ObjSyn = GetSynopticObject
        lRet = ObjSyn.FitInWindow
        MsgBox "FitInWindow = " & CStr(lRet), vbOkOnly, "Test FitInWindow "
        Set ObjSyn = Nothing
End Sub
```

FrameTitle, SynopticCmdTarget Property

Syntax FrameTitle = _String

Description this property sets or returns the name of the frame which loaded the screen. This property has no significance when used from WebClient, and in this case an empty string will be returned.

Parameter	Description
None	None

Result String

Example:

```
'Screen environment
Public Sub Click()
        MsgBox "FrameTitle = " & FrameTitle , vbOkOnly, GetProjectTitle
End Sub
'On screen object environment
Public Sub Click()
        Dim ObjSyn As SynopticCmdTarget
        Set ObjSyn = GetSynopticObject
        MsgBox "FrameTitle = " & ObjSyn.FrameTitle , vbOkOnly, GetProjectTitle
        Set ObjSyn = Nothing
End Sub
```

GlobalContainerName, SynopticCmdTarget Property

Syntax GlobalContainerName = _Boolean

Description When this property is enabled the screen can be made global to each one of its child objects. This means the objects inherit the screens specific functions.

Parameter	Description
None	None

Result Boolean

Example:

```
'Screen scope
Public Sub Click()
        Dim lRet As Integer
        lRet = GlobalContainerName
        MsgBox "GlobalContainerName = " & CStr(lRet), vbOkOnly, "Test GlobalContainerName "
End Sub
'On screen object scope
Public Sub Click()
        Dim ObjSyn As SynopticCmdTarget
        Dim lRet As Long
        Set ObjSyn = GetSynopticObject
        lRet = ObjSyn.GlobalContainerName
        MsgBox "GlobalContainerName = " & CStr(lRet), vbOkOnly, "Test GlobalContainerName "
        Set ObjSyn = Nothing
End Sub
```

GradientColor, SynopticCmdTarget Property

Syntax GradientColor = _Long

Description This property sets or returns the object's Gradient colour. Accepts a Long parameter containing the code for the color shade (R,G,B in each byte). You may find it easier to use the Movicon RGB function.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        GradientColor =RGB(255,255,255)
        sRet = GradientColor
        MsgBox "GradientColor = " & sRet, vbOkOnly, GetProjectTitle 'Return-> 16777215
End Sub
```

GradientFill, SynopticCmdTarget Property

Syntax GradientFill = _Integer

Description This property sets or returns, by using a numeric code, the direct of the screen's back color gradient.
This property can have the following values:

0 = none,
1 = from right to left
2 = from center outwards horizontally
3 = from left to right
4 = from bottom to top
5 = from center outwards vertically

6 = from top to bottom
7 = from the center outwards
8 = From top left corner diagonally
9 = from top right corner diagonally
10 = from bottom right corner diagonally
11 = from top left corner diagonally

Parameter	Description
None	None

Result Integer

Example:

```
'Screen scope
Public Sub Click()
        For i = 0 To 11 Step 1
        GradientFill = i
        sRet = GradientFill
        MsgBox "GradientFill = " & sRet, vbOkOnly, GetProjectTitle
        Next i
End Sub
'On screen object scope
Public Sub Click()
        Dim ObjSyn As SynopticCmdTarget
        Set ObjSyn = GetSynopticObject
        For i = 0 To 11 Step 1
        ObjSyn.GradientFill = i
        sRet = ObjSyn.GradientFill
        MsgBox "GradientFill = " & sRet, vbOkOnly, GetProjectTitle
        Next i
        Set ObjSyn = Nothing
End Sub
```

LayerVariable, SynopticCmdTarget Property

Syntax LayerVariable = _String

Description This property returns the name of the Movicon Real Time variable which determines the number of layers to be displayed for the screen's controls and symbols.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sRet As String
        sRet = LayerVariable
        MsgBox "LayerVariable = " & sRet, vbOkOnly, GetProjectTitle
End Sub
```

MaxInstances, SynopticCmdTarget Property

Syntax MaxInstances = _Integer

Description This property lets you read the value of the screen's maximum number of instances.

Note: The opening of each screen's instance causes the re-reading of the screen description's XML file and therefore the MaxInstances is set to the value defined in it.

Parameter	Description
None	None

Result Integer

Example:

```
Dim objScreen As SynopticCmdTarget
        Set objScreen = GetSynopticObject
        Debug.Print objScreen.MaxInstances()
        Set objScreen = Nothing
End Sub
```

NonDestroyable, SynopticCmdTarget Property

Syntax NonDestroyable = _Integer

Description This option sets or returns the property so that the screen can not be destroyed or rather closed definitively. Therefore, even when not displayed, the screen will continue to occupy the system's memory and resources so that it can be quickly displayed whenever needed.

Parameter	Description
None	None

Result Integer

Example:

```
'Screen scope
Public Sub Click()
        Dim lRet As Integer
        lRet = NonDestroyable
        MsgBox "NonDestroyable = " & CStr(lRet), vbOkOnly, "Test NonDestroyable "
End Sub
'On screen object scope
Public Sub Click()
        Dim ObjSyn As SynopticCmdTarget
```

```
        Dim lRet As Long
        Set ObjSyn = GetSynopticObject
        lRet = ObjSyn.NonDestroyable
        MsgBox "NonDestroyable = " & CStr(lRet), vbOkOnly, "Test NonDestroyable "
        Set ObjSyn = Nothing
End Sub
```

NumColors, SynopticCmdTarget Property

Syntax NumColors = _Integer

Description This property allows you to set or read the number of colours used for managing graphics on the screen. The values are:

0 = Default
1 = Black and White

Setting this property with a new value would not make sense due to the fact that there is not method to update the page with the new values for the time being.

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        If GetSynopticObject.NumColors <> 0 Then
                GetSynopticObject.NumColors = 0
        Else
                GetSynopticObject.NumColors = 1
        End If
End Sub
```

ParameterFileName, SynopticCmdTarget Property

Syntax ParameterFileName = _String

Description This property returns a string containing the screen's parameter file name. The parameter file text must be saved in **UNICODE** format. A screen can be opened in parameter mode using the appropriate commands from the UIInterface. When the screen is parameter driven the variables used in symbols execution properties can be replaced with others specified in the object's text file. The structure must be structured as <original variable>, <new variable>. The parameters have not influence on the screen's objects or on the basic codes contained in the drawings or symbols. An example of a parameter file is described below:

VAR00001,New01
VAR00002,New02
VAR00003,New03

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sRet As String
        sRet = ParameterFileName
        MsgBox "ParameterFileName = " & sRet, vbOkOnly, GetProjectTitle
End Sub
```

ScrollPositionX, SynopticCmdTarget Property

Syntax ScrollPositionX = _Boolean

Description This property sets or returns the screen's scroll's position X. This function has effect only when the screen's scroll bars have been enabled.

This property is not supported in Windows CE.(If used, always returns zero)

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        GetSynopticObject.ScrollPositionX = GetSynopticObject.ScrollPositionX + 5
End Sub
```

ScrollPositionY, SynopticCmdTarget Property

Syntax ScrollPositionY = _Boolean

Description This property sets or returns the screen's scroll's position Y. This function only has effect when the screen's scroll bars have been enabled.

This property is not supported in Windows CE.(If used, always returns a zero)

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
        GetSynopticObject.ScrollPositionY = GetSynopticObject.ScrollPositionY + 5
End Sub
```

SeparateThread, SynopticCmdTarget Property

Syntax SeparateThread = _Boolean

Description When this property is enabled a the screen will be executed in a separate thread, independently from the execution of the other project threads. This function is useful when screens contain significant logic or synapses processing. Logic processing is done in a separate thread without effecting the graphic interface even though a major use of the memory resources are required.

Parameter	Description
None	None

Result Boolean

Example:

```
'Screen scope
Public Sub Click()
        Dim lRet As Integer
        lRet = SeparateThread
        MsgBox "SeparateThread = " & CStr(lRet), vbOkOnly, "Test SeparateThread "
End Sub
'On screen object scope
Public Sub Click()
        Dim ObjSyn As SynopticCmdTarget
        Dim lRet As Long
        Set ObjSyn = GetSynopticObject
        lRet = ObjSyn.SeparateThread
        MsgBox "SeparateThread = " & CStr(lRet), vbOkOnly, "Test SeparateThread "
        Set ObjSyn = Nothing
End Sub
```

ShowOnMDITabsFlag, SynopticCmdTarget Property

Syntax ShowOnMDITabsFlag = _Boolean

Description When this property is enabled the screen's Tab MDI will also be shown in Runtime mode.

This property is not supported in Windows CE.(If set, always returns 'false')

Parameter	Description
None	None

Result Boolean

Example:

```
'Screen scope
Public Sub Click()
        Dim lRet As Integer
        lRet = ShowOnMDITabsFlag
        MsgBox "ShowOnMDITabsFlag = " & CStr(lRet), vbOkOnly, "Test ShowOnMDITabsFlag "
End Sub
'On screen object scope
Public Sub Click()
        Dim ObjSyn As SynopticCmdTarget
        Dim lRet As Long
        Set ObjSyn = GetSynopticObject
        lRet = ObjSyn.ShowOnMDITabsFlag
        MsgBox "ShowOnMDITabsFlag = " & CStr(lRet), vbOkOnly, "Test ShowOnMDITabsFlag "
        Set ObjSyn = Nothing
End Sub
```

SlowTickCounter, SynopticCmdTarget Property

Syntax SlowTickCounter = _Long

Description This property permits you to set or show the number of loops per tick in the pending synapses process or in animations managed on polling associated to the screen. However it would be best not to change the default value. The SlowTickCounter is especially for operations with low priorities.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim sRet As Long
        sRet = SlowTickCounter
        MsgBox "SlowTickCounter = " & sRet, vbOkOnly, GetProjectTitle
End Sub
```

SlowTickFrequency, SynopticCmdTarget Property

Syntax SlowTickFrequency = _Long

Description This property allows you to set or show the Tick Frequency for managing animation on polling and for processing logic of the synapses associated to the screen. It is advisable not to change the default value. The SlowTickFrequency is especially for operations with lower priorities.

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
        Dim sRet As Long
        sRet = SlowTickFrequency
        MsgBox "SlowTickFrequency = " & sRet, vbOkOnly, GetProjectTitle
End Sub
```

SynapsisExecution, SynopticCmdTarget Property

Syntax SynapsisExecution = _Boolean

Description This property sets or returns the execution status of the synapses on screen. When this property is set at True, the synapses logic contained on screen will be run. The false value is returned when execution run has been completed.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub SynopticLoading()
        ' Run the synapsis
        SynapsisExecution = True
End Sub
```

SynopticHeight, SynopticCmdTarget Property

Syntax SynopticHeight = _Integer

Description This property sets or returns the screen's height.

Parameter	Description
None	None

Result Integer

Example:

```
'Screen scope
Public Sub Click()
        MsgBox "SynopticHeight = " & SynopticHeight , vbOkOnly, GetProjectTitle
End Sub
'On screen object scope
Public Sub Click()
        Dim ObjSyn As SynopticCmdTarget
        Set ObjSyn = GetSynopticObject
        MsgBox "SynopticHeight = " & ObjSyn.SynopticHeight , vbOkOnly, GetProjectTitle
        Set ObjSyn = Nothing
End Sub
```

SynopticID, SynopticCmdTarget Property

Syntax SynopticID = _Long

Description This property sets or returns the screen's ID.

Parameter	Description
None	None

Result Long

Example:

```
'Screen scope
Public Sub Click()
        MsgBox "FrameTitle = " & SynopticID , vbOkOnly, GetProjectTitle
End Sub
'On screen object scope
Public Sub Click()
        Dim ObjSyn As SynopticCmdTarget
        Set ObjSyn = GetSynopticObject
        MsgBox "FrameTitle = " & ObjSyn.SynopticID , vbOkOnly, GetProjectTitle
        Set ObjSyn = Nothing
End Sub
```

SynopticPublicSource, SynopticCmdTarget Property

Syntax SynopticPublicSource = _String

Description This property returns the name of the reference screen for the Public symbols. When symbols have been inserted on screen with the same "Public Name", and one of these exists in the "Public Symbol Container" Screen, they will be changed with the reference symbol's properties when the Runtime mode starts.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sRet As String
        sRet = SynopticPublicSource
        MsgBox "SynopticPublicSource = " & sRet, vbOkOnly, GetProjectTitle
End Sub
```

SynopticWidth, SynopticCmdTarget Property

Syntax SynopticWidth = _Integer

Description This property sets or returns the screen width.

Parameter	Description
None	None

Result Integer

Example:

```
'Screen scope
Public Sub Click()
        MsgBox "SynopticWidth = " & SynopticWidth, vbOkOnly, GetProjectTitle
End Sub
'On screen object scope
Public Sub Click()
        Dim ObjSyn As SynopticCmdTarget
        Set ObjSyn = GetSynopticObject
        MsgBox "SynopticWidth = " & ObjSyn.SynopticWidth, vbOkOnly, GetProjectTitle
        Set ObjSyn = Nothing
End Sub
```

UseAntialaising, SynopticCmdTarget Property

Syntax UseAntialaising = _Boolean

Description This property activates or deactivates the graphical Anti Alaising in drawings belong to the screen.
Caution: the anti aliasing has no effect in Windows CE systems.

Caution: the anti aliasing has no effect in Windows CE systems.

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
Dim oSyn As SynopticCmdTarget
        Set oSyn = GetSynopticObject
        oSyn.UseAntialiasing = Not(oSyn.UseAntialiasing)
End Sub
```

XBackImage, SynopticCmdTarget Property

Syntax XBackImage = _Integer

Description This property sets or returns the value relating to the coordinate on the x axis of the image on the screen.

Parameter	Description
None	None

Result Integer

Example:

```
'Screen scope
Public Sub Click()
        Dim lRet As Integer
        lRet = XBackImage
        MsgBox "XBackImage = " & CStr(lRet), vbOkOnly, "Test XBackImage "
        XBackImage = 100
        lRet = XBackImage
        MsgBox "XBackImage = " & CStr(lRet), vbOkOnly, "Test XBackImage "
End Sub
'On screen object scope
Public Sub Click()
        Dim ObjSyn As SynopticCmdTarget
```

```
        Dim lRet As Long
        Set ObjSyn = GetSynopticObject
        lRet = ObjSyn.XBackImage
        MsgBox "XBackImage = " & CStr(lRet), vbOkOnly, "Test XBackImage "
        ObjSyn.XBackImage = 100
        lRet = ObjSyn.XBackImage
        MsgBox "XBackImage = " & CStr(lRet), vbOkOnly, "Test XBackImage "
        Set ObjSyn = Nothing
End Sub
```

YBackImage, SynopticCmdTarget Property

Syntax YBackImage = _Integer

Description This property sets or returns the value relating to the coordinate on the y axis of the image on screen.

Parameter	Description
None	None

Result Integer

Example:

```
'Screen scope
Public Sub Click()
        Dim lRet As Integer
        lRet = YBackImage
        MsgBox "YBackImage = " & CStr(lRet), vbOkOnly, "Test YBackImage "
        YBackImage = 100
        lRet = YBackImage
        MsgBox "YBackImage = " & CStr(lRet), vbOkOnly, "Test YBackImage "
End Sub
'On screen object scope
Public Sub Click()
        Dim ObjSyn As SynopticCmdTarget
        Dim lRet As Long
        Set ObjSyn = GetSynopticObject
        lRet = ObjSyn.YBackImage
        MsgBox "YBackImage = " & CStr(lRet), vbOkOnly, "Test YBackImage "
        ObjSyn.YBackImage = 100
        lRet = ObjSyn.YBackImage
        MsgBox "YBackImage = " & CStr(lRet), vbOkOnly, "Test YBackImage "
        Set ObjSyn = Nothing
End Sub
```

ZoomFactorX, SynopticCmdTarget Property

Syntax ZoomFactorX = _Double

Description This property sets or returns the Zoom along the x axis of the screen within the window.

Parameter	Description
None	None

Result Double

Example:

```
Public Sub Click()
        Dim sRet As Double
        sRet = ZoomFactorX
        MsgBox "ZoomFactorX = " & sRet, vbOkOnly, GetProjectTitle
        ZoomFactorX = 0.5
        sRet = ZoomFactorX
        MsgBox "ZoomFactorX = " & sRet, vbOkOnly, GetProjectTitle
End Sub
```

ZoomFactorY, SynopticCmdTarget Property

Syntax ZoomFactorY = _Double

Description This property returns or sets the zoom value along the Y axis of the screen within the window.

Parameter	Description
None	None

Result Double

Example:

```
Public Sub Click()
        Dim sRet As Double
        sRet = ZoomFactorY
        MsgBox "ZoomFactorY = " & sRet, vbOkOnly, GetProjectTitle
        ZoomFactorY = 0.5
        sRet = ZoomFactorY
        MsgBox "ZoomFactorY = " & sRet, vbOkOnly, GetProjectTitle
End Sub
```

53. TraceDBWndCmdTarget

OnFilter, TraceDBWndCmdTarget Event

Description Event occurs each time a request is made to apply a filter for extracting data the trace file.

Parameter	Description
bRet As Boolean	Enable on status change.

OnPrint, TraceDBWndCmdTarget Event

Description Event occurs each time a request is made to print data loaded in the display window.

This event is not supported in Windows CE.

Parameter	Description
bRet As Boolean	Enabled at print startup.

OnRefresh, TraceDBWndCmdTarget Event

Description Event occurs each time a request is made to refresh data loaded in the display window.

Parameter	Description
bRet As Boolean	Enable on status change.

EditCopy, TraceDBWndCmdTarget Function

Syntax EditCopy()

Description This property copies the selected line contents onto the clipboard.

Parameter	Description

None	None

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
	Dim TraceDBWnd As TraceDBWndCmdTarget
	Set TraceDBWnd = GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
	If Not TraceDBWnd Is Nothing Then
		TraceDBWnd.EditCopy
	End If
	Set TraceDBWnd = Nothing
End Sub
```

EditLayout, TraceDBWndCmdTarget Function

Syntax EditLayout()

Description This function opens the configuration window of fields to be displayed in the Trace window.

This function is only executed if the "Show Control window" property has been enabled in the Window object. Otherwise the "Field Choice Window" will not open and this function will return the "False" value.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
	Dim TraceDBWnd As TraceDBWndCmdTarget
	Set TraceDBWnd = GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
	If Not TraceDBWnd Is Nothing Then
		TraceDBWnd.EditLayout
	End If
	Set TraceDBWnd = Nothing
End Sub
```

LoadExtSettings, TraceDBWndCmdTarget Function

Syntax LoadExtSettings

Description This function permits the object's relating external file settings to be loaded. This file can be specified in design mode in the "External File settings" property or in

the "ExtSettingsFile" interface properties. The extension provided for this file is ".SXML".

Parameter	Description
None	None

Result Boolean

Example:

```
Public Sub Click()
Dim objSymbol As TraceDBWndCmdTarget
Set objSymbol = GetSynopticObject.GetSubObject("TestObject").GetObjectInterface
If objSymbol Is Nothing Then Exit Sub
objSymbol.ExtSettingsFile = "test.sxml"
objSymbol.LoadExtSettings
Set objSymbol = Nothing
End Sub
```

RecalcLayout, TraceDBWndCmdTarget Function

Syntax RecalcLayout()

Description The function updates the object graphical layout. This function needs to be executed after a property involving the object's graphical aspect, has been edited such as changing the sizes of one of the columns.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
        Dim TraceDBWnd As TraceDBWndCmdTarget
        Set TraceDBWnd = GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
        If Not TraceDBWnd Is Nothing Then
                TraceDBWnd .AutoLayout = Not TraceDBWnd .AutoLayout
                TraceDBWnd .RecalcLayout
        End If
        Set TraceDBWnd = Nothing
End Sub
```

Refresh, TraceDBWndCmdTarget Function

Syntax Refresh()

Description This function refreshes the data in the object which is useful when the query for extracting data from the TraceDB is edited.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
        Dim TraceDBWnd As TraceDBWndCmdTarget
        Set TraceDBWnd = GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
        If Not TraceDBWnd Is Nothing Then
                TraceDBWnd .Query = "SELECT * FROM VAR00001 ORDER BY VAR00001.TimeCol
                DESC"
                TraceDBWnd .Refresh
        End If
        Set TraceDBWnd = Nothing
End Sub
```

SaveExtSettings, TraceDBWndCmdTarget Function

Syntax SaveExtSettings

Description This function permits the objects settings to be save in the relating external settings file. This file can be specified when in design mode in the "Ext. Settings File" property, or using the property from the "ExtSettingsFile" interface. The extension provided for this file is ".SXML".

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
Dim objSymbol As TraceDBWndCmdTarget
Set objSymbol = GetSynopticObject.GetSubObject("TestObject").GetObjectInterface
If objSymbol Is Nothing Then Exit Sub
objSymbol.ExtSettingsFile = "test.sxml"
objSymbol.SaveExtSettings
Set objSymbol = Nothing
End Sub
```

AutoLayout, TraceDBWndCmdTarget Property

Syntax AutoLayout = _Boolean

Description When enabling this property, the layout will be set to automatic mode. This means that the columns will be automatically resized so that they all fit into the area of the TraceDB Window. When this property is disabled,

the columns will show with the sizes set during programming mode when the window is opened. The last columns, on the right, may not fit into the window and will have to be viewed by using the horizontal scroll bar.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
	Dim TraceDBWnd As TraceDBWndCmdTarget
	Set TraceDBWnd = GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
	If Not TraceDBWnd Is Nothing Then
		TraceDBWnd.AutoLayout = Not TraceDBWnd.AutoLayout
		TraceDBWnd.RecalcLayout
	End If
	Set TraceDBWnd = Nothing
End Sub
```

ButtonPos, TraceDBWndCmdTarget Property

Syntax ButtonPos = _Integer

Description This setting returns the position where the buttons are to appear in the Trace window.

The options are:
0 = left
1 = top
2 = right
3 = below

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
	Dim objWnd As TraceDBWndCmdTarget
	Set objWnd = GetSynopticObject.GetSubObject("TraceLog").GetObjectInterface
	If Not objWnd Is Nothing Then
		MsgBox "objWnd 's ButtonPos is " & objWnd .ButtonPos,vbInformation,GetProjectTitle
		objWnd .ButtonPos = 2
		objWnd .RecalcLayout
	Else
		MsgBox "objWnd is nothing",vbInformation,GetProjectTitle
	End If
End Sub
```

ButtonSize, TraceDBWndCmdTarget Property

Syntax ButtonSize = _Integer

Description This setting returns the size of the buttons which are to be displayed in the Trace window.

The options are:
0 = small
1 = medium
2 = large

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
	Dim objWnd As TraceDBWndCmdTarget
	Set objWnd = GetSynopticObject.GetSubObject("TraceLog").GetObjectInterface
	If Not objWnd Is Nothing Then
		MsgBox "objWnd 's ButtonSize is " & objWnd .ButtonSize,vbInformation,GetProjectTitle
		objWnd .ButtonSize= 2
		objWnd .RecalcLayout
	Else
		MsgBox "objWnd is nothing",vbInformation,GetProjectTitle
	End If
End Sub
```

Clickable, TraceDBWndCmdTarget Property

Syntax Clickable = _Boolean

Description This property is used to define whether the operator can interact with the Trace window. When this property is disabled, the control will no longer respond when either clicked by the mouse or operated from keyboard.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
```

```
        Dim TraceDBWnd As TraceDBWndCmdTarget
        Set TraceDBWnd = GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
        If Not TraceDBWnd Is Nothing Then
                TraceDBWnd.Clickable = Not TraceDBWnd.Clickable
                TraceDBWnd.RecalcLayout
        End If
        Set TraceDBWnd = Nothing
End Sub
```

ExtSettingsFile, TraceDBWndCmdTarget Property

Syntax ExtSettingsFile = _String

Description This property sets or returns the external configuration file for the referenced object. the file can be alsow specified in design mode in the object's "Configuration File" property. The extension provided for this file is ".SXML".

Parameter	Description
None	None

Result Long

Example:

```
Public Sub Click()
Dim objSymbol As TraceDBWndCmdTarget
Set objSymbol = GetSynopticObject.GetSubObject("TestObject").GetObjectInterface
If objSymbol Is Nothing Then Exit Sub
objSymbol.ExtSettingsFile = "test.sxml"
objSymbol.SaveExtSettings
Set objSymbol= Nothing
End Sub
```

FileReport, TraceDBWndCmdTarget Property

Syntax FileReport = _String

Description This property sets or returns the report file name to be used for printing data displayed in the Variable Trace window. This file must be created with the Report Designer or Crystal Report© (.rpt). If this field is left empty, Movicon will us the default report file created by Progea in the Report Designer format.

This property is not supported in Windows CE.(If set, always returns an empty string)

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
	Dim objWnd As TraceDBWndCmdTarget
	Set objWnd = GetSynopticObject.GetSubObject("TraceLog").GetObjectInterface
	If Not objWnd Is Nothing Then
		MsgBox "objWnd 's FileReport is " & objWnd.FileReport ,vbInformation,GetProjectTitle
		objWnd.FileReport = "C:\Report1.rpt"
		objWnd.RecalcLayout
	Else
		MsgBox "objWnd is nothing",vbInformation,GetProjectTitle
	End If
End Sub
```

FilterBtnText, TraceDBWndCmdTarget Property

Syntax FilterBtnText = _String

Description This property sets or returns a text for the command button used for filtering data in the Trace window. When nothing is specified, Movicon will use the default text.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
	Dim objWnd As TraceDBWndCmdTarget
	Set objWnd = GetSynopticObject.GetSubObject("TraceLog").GetObjectInterface
	If Not objWnd Is Nothing Then
		MsgBox "objWnd 's FilterBtnText is " & objWnd.FilterBtnText,vbInformation,GetProjectTitle
		objWnd.FilterBtnText = "Filter options"
		objWnd.RecalcLayout
	Else
		MsgBox "objWnd is nothing",vbInformation,GetProjectTitle
	End If
End Sub
```

FilterFromDate, TraceDBWndCmdTarget Property

Syntax FilterFromDate = _Date

Description This property sets or returns the 'From Date Filter' for displaying messages in the Movicon Trace window.

Parameter	Description
None	None

Result Date

Example:

```
Option Explicit
Public Sub Click()
	Dim TraceDBWnd As TraceDBWndCmdTarget
	Set TraceDBWnd = GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
	If Not TraceDBWnd Is Nothing Then
		MsgBox "TraceDBWnd's FilterFromDate is " & TraceDBWnd .FilterFromDate,vbInformation,GetProjectTitle
		TraceDBWnd .FilterFromDate = Now()
		TraceDBWnd .Refresh
	Else
		MsgBox "TraceDBWnd is nothing",vbInformation,GetProjectTitle
	End If
	Set TraceDBWnd = Nothing

End Sub
```

FilterToDate, TraceDBWndCmdTarget Property

Syntax FilterToDate = _Date

Description This property sets or returns the 'To Date Filter' for displaying messages in the Movicon Historical Log window.

Parameter	Description
None	None

Result Date

Example:

```
Option Explicit
Public Sub Click()
	Dim TraceDBWnd As TraceDBWndCmdTarget
	Set TraceDBWnd = GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
	If Not TraceDBWnd Is Nothing Then
		MsgBox "TraceDBWnd's FilterToDate is " & TraceDBWnd .FilterToDate,vbInformation,GetProjectTitle
		TraceDBWnd .FilterToDate = Now()
		TraceDBWnd .Refresh
	Else
		MsgBox "TraceDBWnd is nothing",vbInformation,GetProjectTitle
	End If
```

```
        Set TraceDBWnd = Nothing

End Sub
```

FilterUser, TraceDBWndCmdTarget Property

Syntax FilterUser = _String

Description This property sets or returns the 'User Filter' for displaying messages in the Movicon Trace window.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim TraceDBWnd As TraceDBWndCmdTarget
        Set TraceDBWnd = GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
        If Not TraceDBWnd Is Nothing Then
                MsgBox "TraceDBWnd's FilterUser is " & TraceDBWnd.FilterUser,vbInformation,GetProjectTitle
                TraceDBWnd.FilterUser = "User00001"
                TraceDBWnd.Refresh
        Else
                MsgBox "TraceDBWnd is nothing",vbInformation,GetProjectTitle
        End If
        Set TraceDBWnd = Nothing

End Sub
```

GraphicButtons, TraceDBWndCmdTarget Property

Syntax GraphicButtons = _Boolean

Description When Enabling this property, the TraceDB Window buttons are drawn using an icon instead of text. The text will instead be displayed as a tooltip when positioning the mouse on top of the button.

The tooltip is not managed in Windows CE versions.

This property is not managed by the 'Alarm Banner' object.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Click()
        GraphicButtons = True
        RecalcLayout
End Sub
```

IncludeMilliseconds, TraceDBWndCmdTarget Property

Syntax IncludeMilliseconds = _Boolean

Description This property allows you to define whether or not to display the milliseconds in the window's time column.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
        Dim TraceDBWnd As TraceDBWndCmdTarget
        Set TraceDBWnd = GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
        If Not TraceDBWnd Is Nothing Then
                TraceDBWnd.IncludeMilliseconds = Not TraceDBWnd.IncludeMilliseconds
                TraceDBWnd.Refresh
        End If
        Set TraceDBWnd = Nothing
End Sub
```

MaxCount, TraceDBWndCmdTarget Property

Syntax MaxCount = _Integer

Description This property allows you to set the maximum number of rows to be displayed in the "TraceDB Window".

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
```

```
Public Sub Click()
        Dim TraceDBWnd As TraceDBWndCmdTarget
        Set TraceDBWnd = GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
        If Not TraceDBWnd Is Nothing Then
                TraceDBWnd.MaxCount = 150
                TraceDBWnd.Refresh
        End If
        Set TraceDBWnd = Nothing
End Sub
```

NetworkBackupServerName, TraceDBWndCmdTarget Property

Syntax NetworkBackupServerName = _String

Description This property sets or returns the name of any Network Backup Server used for getting data to display in the TraceDB Window when the primary server, the one set in the 'NetowrkServerName'property is in timeout.

Parameter	Description
None	None

Result String

Example:

```
Dim TraceDBWnd As TraceDBWndCmdTarget

Public Sub Click()
        Debug.Print objTraceDBWnd.NetworkBackupServerName
End Sub
Public Sub SymbolLoading()
        Set objTraceDBWnd=
        GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
End Sub
```

NetworkServerName, TraceDBWndCmdTarget Property

Syntax NetworkServerName = _String

Description This property returns the name of any Network Server where data is to be retrieved for displaying in the TraceDB window.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim TraceDBWnd As TraceDBWndCmdTarget
        Set TraceDBWnd =
        GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
        If Not TraceDBWnd Is Nothing Then
                TraceDBWnd.NetworkServerName = "PERSONAL11"
                TraceDBWnd.Refresh
        End If
        Set TraceDBWnd = Nothing
End Sub
```

PrintBtnText, TraceDBWndCmdTarget Property

Syntax PrintBtnText = _String

Description This property sets or returns a text for the command button form printing the data displayed in the Trace window. When nothing is entered Movicon will use the default text.

This property is not supported in Windows CE.(If set, always returns an empty string)

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim objWnd As TraceDBWndCmdTarget
        Set objWnd = GetSynopticObject.GetSubObject("TraceLog").GetObjectInterface
        If Not objWnd Is Nothing Then
                MsgBox "objWnd 's PrintBtnText is " & objWnd.PrintBtnText
                ,vbInformation,GetProjectTitle
                objWnd.PrintBtnText = "Print data"
                objWnd.RecalcLayout
        Else
                MsgBox "objWnd is nothing",vbInformation,GetProjectTitle
        End If
End Sub
```

Project, TraceDBWndCmdTarget Property

Syntax Project = _String

Description This property allows you to set the name of the child project from which you wish to recuperate data to be displayed. The current project will be used if this field is left blank.

The name of the eventual child project of the current project is to be inserted exclusively.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim TraceDBWnd As TraceDBWndCmdTarget
        Set TraceDBWnd = GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
        If Not TraceDBWnd Is Nothing Then
                Debug.Print TraceDBWnd .Project
        End If
        Set TraceDBWnd = Nothing
End Sub
```

Query, TraceDBWndCmdTarget Property

Syntax Query = _String

Description This property allows you to set a selection Query in SQL language for extracting data contained in the TraceDB. This query is executed for default upon each data refresh in the window, whether executed automatically or on the operator's command.

A table is created in the database for each variable enabled in the "Trace".

The tables are structured with the following columns:

ActionCol	Action
ValueCol	Value
TimeCol	Event Time
UserCol	User
BeforeCol	Before
AfterCol	After
QualityCol	Quality

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim TraceDBWnd As TraceDBWndCmdTarget
        Set TraceDBWnd = GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
        If Not TraceDBWnd Is Nothing Then
```

```
            TraceDBWnd .Query = "SELECT * FROM VAR00001 ORDER BY VAR00001.TimeCol
            DESC"
            TraceDBWnd .Refresh
        End If
        Set TraceDBWnd = Nothing
End Sub
```

RefreshBtnText, TraceDBWndCmdTarget Property

Syntax RefreshBtnText = _String

Description This property sets or returns a text for the command button which refreshes data displayed in the Trace window. When nothing is specified, Movicon will use the default text.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim objWnd As TraceDBWndCmdTarget
        Set objWnd = GetSynopticObject.GetSubObject("TraceLog").GetObjectInterface
        If Not objWnd Is Nothing Then
            MsgBox "objWnd 's RefreshBtnText is " &
            objWnd.RefreshBtnText,vbInformation,GetProjectTitle
            objWnd.RefreshBtnText = "Refresh all"
            objWnd.RecalcLayout
        Else
            MsgBox "objWnd is nothing",vbInformation,GetProjectTitle
        End If
End Sub
```

ShowFilterBtn, TraceDBWndCmdTarget Property

Syntax ShowFilterBtn = _Boolean

Description This property allows you to display the command button for filtering data in the Trace window.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
        Dim objTraceWnd As HisLogWndCmdTarget
        Set objTraceWnd = GetSynopticObject.GetSubObject("TraceLog").GetObjectInterface
        If Not objTraceWnd Is Nothing Then
                objTraceWnd .ShowFilterBtn = Not objTraceWnd .ShowFilterBtn
                objTraceWnd .RecalcLayout
        End If
        Set objTraceWnd = Nothing
End Sub
```

ShowPrintBtn, TraceDBWndCmdTarget Property

Syntax ShowPrintBtn = _Boolean

Description This property allows the command button to be shown for printing data from the Trace window.
The print is executed using the report fiel which should be specified in the "Report File" property. Movicon passes the same filter settings to the report for printing data which coincide with the data shown in the window in question.

This property is not supported in Windows CE.(If set, always returns 'false')

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
        Dim objTraceWnd As HisLogWndCmdTarget
        Set objTraceWnd = GetSynopticObject.GetSubObject("TraceLog").GetObjectInterface
        If Not objTraceWnd Is Nothing Then
                objTraceWnd .ShowPrintBtn = Not objTraceWnd .ShowPrintBtn
                objTraceWnd .RecalcLayout
        End If
        Set objTraceWnd = Nothing
End Sub
```

ShowRefreshBtn, TraceDBWndCmdTarget Property

Syntax ShowRefreshBtn = _Boolean

Description This property allows you to show the command button for refreshing data in the Trace window.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
	Dim objTraceWnd As HisLogWndCmdTarget
	Set objTraceWnd = GetSynopticObject.GetSubObject("TraceLog").GetObjectInterface
	If Not objTraceWnd Is Nothing Then
		objTraceWnd .ShowRefreshBtn = Not objTraceWnd .ShowRefreshBtn
		objTraceWnd .RecalcLayout
	End If
	Set objTraceWnd = Nothing
End Sub
```

SortBy, TraceDBWndCmdTarget Property

Syntax SortBy = _String

Description This property sets or returns the 'Sort By' filter for displaying messages in the Movicon Trace window.

ActionCol
AfterCol
BeforeCol
LocalCol
MSecCol
QualityCol
TimeCol
UserCol
ValueCol

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
	Dim objTraceWnd As TraceDBWndCmdTarget
	Set objTraceWnd = GetSynopticObject.GetSubObject("TraceLog").GetObjectInterface
	If Not objTraceWnd Is Nothing Then
		MsgBox "objTraceWnd 's SortBy is " & objTraceWnd .SortBy
		,vbInformation,GetProjectTitle
		objTraceWnd .SortBy = "ActionCol"
		objTraceWnd .Refresh
	Else
		MsgBox "objTraceWnd is nothing",vbInformation,GetProjectTitle
	End If
End Sub
```

SubItemAction, TraceDBWndCmdTarget Property

Syntax SubItemAction = _String

Description Allows you to set the text which is to appear as the "Action" column's name. The default text will be used when this field is left blank.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim TraceDBWnd As TraceDBWndCmdTarget
        Set TraceDBWnd = GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
        If Not TraceDBWnd Is Nothing Then
                TraceDBWnd .SubItemAction = "Type action"
                TraceDBWnd .RecalcLayout
        End If
        Set TraceDBWnd = Nothing
End Sub
```

SubItemActionPos, TraceDBWndCmdTarget Property

Syntax SubItemActionPos = _Integer

Description This property sets or returns the position of the "Acton" column within the Trace DB Window. When setting a new value, the other columns will be automatically re-positioned in the window layout. In addition when setting the "-1", the column will be hidden. The "0" value is used to indicate position of the first column on the left in the window.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
        Dim TraceDBWnd As TraceDBWndCmdTarget
        Set TraceDBWnd = GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
        If Not TraceDBWnd Is Nothing Then
```

```
            Debug.Print TraceDBWnd .SubItemActionPos
        End If
        Set TraceDBWnd = Nothing
End Sub
```

SubItemActionWidth, TraceDBWndCmdTarget Property

Syntax SubItemActionWidth = _Integer

Description This property indicates the size in pixels of the column in the Trace display window. The -1 value is returned when the column is not displayed.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
        Dim TraceDBWnd As TraceDBWndCmdTarget
        Set TraceDBWnd = GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
        If Not TraceDBWnd Is Nothing Then
            TraceDBWnd .SubItemActionWidth = 20
            TraceDBWnd .RecalcLayout
        End If
        Set TraceDBWnd = Nothing
End Sub
```

SubItemAfter, TraceDBWndCmdTarget Property

Syntax SubItemAfter = _String

Description Allows you to set the text which is to appear as the "After" column's name. The default text will be used if this field is left blank.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim TraceDBWnd As TraceDBWndCmdTarget
        Set TraceDBWnd = GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
```

```
        If Not TraceDBWnd Is Nothing Then
                TraceDBWnd .SubItemAfter = "After that"
                TraceDBWnd .RecalcLayout
        End If
        Set TraceDBWnd = Nothing
End Sub
```

SubItemAfterPos, TraceDBWndCmdTarget Property

Syntax SubItemAfterPos = _Integer

Description This property sets or returns the position of the "After" column within the Trace DB Window. When setting a new value, the other columns will be automatically re-positioned in the window layout. In addition when setting the "-1", the column will be hidden. The "0" value is used to indicate position of the first column on the left in the window.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
        Dim TraceDBWnd As TraceDBWndCmdTarget
        Set TraceDBWnd = GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
        If Not TraceDBWnd Is Nothing Then
                Debug.Print TraceDBWnd .SubItemAfterPos
        End If
        Set TraceDBWnd = Nothing
End Sub
```

SubItemAfterWidth, TraceDBWndCmdTarget Property

Syntax SubItemAfterWidth = _Integer

Description This property indicates the size in pixels of the column in the Trace display window. The -1 value is returned when the column is not displayed.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
```

```
Public Sub Click()
        Dim TraceDBWnd As TraceDBWndCmdTarget
        Set TraceDBWnd = GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
        If Not TraceDBWnd Is Nothing Then
                TraceDBWnd .SubItemAfterWidth = 20
                TraceDBWnd .RecalcLayout
        End If
        Set TraceDBWnd = Nothing
End Sub
```

SubItemBefore, TraceDBWndCmdTarget Property

Syntax SubItemBefore = _String

Description Allows you to set the text which is to appear as the "Before" column's name. The default text will be used if this field is left blank.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim TraceDBWnd As TraceDBWndCmdTarget
        Set TraceDBWnd = GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
        If Not TraceDBWnd Is Nothing Then
                TraceDBWnd .SubItemBefore = "Before that"
                TraceDBWnd .RecalcLayout
        End If
        Set TraceDBWnd = Nothing
End Sub
```

SubItemBeforePos, TraceDBWndCmdTarget Property

Syntax SubItemBeforePos = _Integer

Description This property sets or returns the position of the "Before" column within the Trace DB Window. When setting a new value, the other columns will be automatically re-positioned in the window layout. In addition when setting the "-1", the column will be hidden. The "0" value is used to indicate position of the first column on the left in the window.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
	Dim TraceDBWnd As TraceDBWndCmdTarget
	Set TraceDBWnd = GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
	If Not TraceDBWnd Is Nothing Then
		Debug.Print TraceDBWnd .SubItemBeforePos
	End If
	Set TraceDBWnd = Nothing
End Sub
```

SubItemBeforeWidth, TraceDBWndCmdTarget Property

Syntax SubItemBeforeWidth = _Integer

Description This property indicates the size in pixels of the column in the Trace display window. The -1 value is returned when the column is not displayed.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
	Dim TraceDBWnd As TraceDBWndCmdTarget
	Set TraceDBWnd = GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
	If Not TraceDBWnd Is Nothing Then
		TraceDBWnd .SubItemBeforeWidth = 20
		TraceDBWnd .RecalcLayout
	End If
	Set TraceDBWnd = Nothing
End Sub
```

SubItemQuality, TraceDBWndCmdTarget Property

Syntax SubItemQuality = _String

Description Allows you to set the text which is to appear as the "Quality" column's name. The default text will be used if this field is left blank.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim TraceDBWnd As TraceDBWndCmdTarget
        Set TraceDBWnd = GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
        If Not TraceDBWnd Is Nothing Then
                TraceDBWnd .SubItemQuality = "With Quality"
                TraceDBWnd .RecalcLayout
        End If
        Set TraceDBWnd = Nothing
End Sub
```

SubItemQualityPos, TraceDBWndCmdTarget Property

Syntax SubItemQualityPos = _Integer

Description This property sets or returns the position of the "Quality" column within the Trace DB window. When setting a new value, the other columns will be automatically re-positioned in the window layout. In addition when setting the "-1", the column will be hidden. The "0" value is used to indicate position of the first column on the left in the window.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
        Dim TraceDBWnd As TraceDBWndCmdTarget
        Set TraceDBWnd = GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
        If Not TraceDBWnd Is Nothing Then
                Debug.Print TraceDBWnd .SubItemQualityPos
        End If
        Set TraceDBWnd = Nothing
End Sub
```

SubItemQualityWidth, TraceDBWndCmdTarget Property

Syntax SubItemQualityWidth = _Integer

Description This property indicates the size in pixels of the column in the Trace display window. The -1 value is returned when the column is not displayed.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
        Dim TraceDBWnd As TraceDBWndCmdTarget
        Set TraceDBWnd = GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
        If Not TraceDBWnd Is Nothing Then
                TraceDBWnd .SubItemQualityWidth = 20
                TraceDBWnd .RecalcLayout
        End If
        Set TraceDBWnd = Nothing
End Sub
```

SubItemTime, TraceDBWndCmdTarget Property

Syntax SubItemTime = _String

Description Allows you to set the text which is to appear as the "Time" column's name. The default text will be used if this field is left blank.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim TraceDBWnd As TraceDBWndCmdTarget
        Set TraceDBWnd = GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
        If Not TraceDBWnd Is Nothing Then
                TraceDBWnd .SubItemTime = "At Time"
                TraceDBWnd .RecalcLayout
        End If
        Set TraceDBWnd = Nothing
End Sub
```

SubItemTimePos, TraceDBWndCmdTarget Property

Syntax SubItemTimePos = _Integer

Description This property sets or returns the position of the "Time" column within the Trace DB window. When setting a new value, the other columns will be automatically re-positioned in the window layout. In addition when setting the "-1", the column will be hidden. The "0" value is used to indicate position of the first column on the left in the window.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
        Dim TraceDBWnd As TraceDBWndCmdTarget
        Set TraceDBWnd = GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
        If Not TraceDBWnd Is Nothing Then
                Debug.Print TraceDBWnd .SubItemTimePos
        End If
        Set TraceDBWnd = Nothing
End Sub
```

SubItemTimeStamp, TraceDBWndCmdTarget Property

Syntax SubItemTimeStamp = _String

Description Here you can enter text to show as the name for the "TimeStamp" Column. The default field will be used if this field is left empty.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim TraceDBWnd As TraceDBWndCmdTarget
        Set TraceDBWnd =
        GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
        If Not TraceDBWnd Is Nothing Then
                TraceDBWnd.SubItemTimeStamp = "Tag TimeStamp"
                TraceDBWnd.RecalcLayout
        End If
        Set TraceDBWnd = Nothing
End Sub
```

SubItemTimeStampPos, TraceDBWndCmdTarget Property

Syntax SubItemTimeStampPos = _Integer

Description This property sets or returns the position of the "TimeStamp" column within the TraceDB Window. when setting a new value the other columns will automatically reposition within the window layout. In addition, when setting value to "-1" , the column will be hidden. the "0" value indicates the position of the first column to the left of the window.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
        Dim TraceDBWnd As TraceDBWndCmdTarget
        Set TraceDBWnd =
        GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
        If Not TraceDBWnd Is Nothing Then
            Debug.Print TraceDBWnd.SubItemTimeStampPos
        End If
        Set TraceDBWnd = Nothing
End Sub
```

SubItemTimeStampWidth, TraceDBWndCmdTarget Property

Syntax SubItemTimeStampWidth = _Integer

Description This property indicates the size of the "TimeStamp" column in pixels within the Trace window. The -1 value will return if this column is not displayed in window.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
        Dim TraceDBWnd As TraceDBWndCmdTarget
        Set TraceDBWnd =
        GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
        If Not TraceDBWnd Is Nothing Then
            TraceDBWnd.SubItemTimeStampWidth = 20
            TraceDBWnd.RecalcLayout
        End If
```

```
        Set TraceDBWnd = Nothing
End Sub
```

SubItemTimeWidth, TraceDBWndCmdTarget Property

Syntax SubItemTimeWidth = _Integer

Description This property indicates the size in pixels of the column in the Trace display window. The -1 value is returned when the column is not displayed.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
        Dim TraceDBWnd As TraceDBWndCmdTarget
        Set TraceDBWnd = GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
        If Not TraceDBWnd Is Nothing Then
                TraceDBWnd .SubItemTimeWidth = 20
                TraceDBWnd .RecalcLayout
        End If
        Set TraceDBWnd = Nothing
End Sub
```

SubItemUser, TraceDBWndCmdTarget Property

Syntax SubItemUser = _String

Description Allows you to set the text which is to appear as the "User" column's name. The default text will be used if this field is left blank.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim TraceDBWnd As TraceDBWndCmdTarget
        Set TraceDBWnd = GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
        If Not TraceDBWnd Is Nothing Then
                TraceDBWnd .SubItemUser = "Logon User"
```

```
            TraceDBWnd .RecalcLayout
        End If
        Set TraceDBWnd = Nothing
End Sub
```

SubItemUserPos, TraceDBWndCmdTarget Property

Syntax SubItemUserPos = _Integer

Description This property sets or returns the position of the "User" column within the Trace DB window. When setting a new value, the other columns will be automatically re-positioned in the window layout. In addition when setting the "-1", the column will be hidden. The "0" value is used to indicate position of the first column on the left in the window.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
        Dim TraceDBWnd As TraceDBWndCmdTarget
        Set TraceDBWnd = GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
        If Not TraceDBWnd Is Nothing Then
            Debug.Print TraceDBWnd .SubItemUserPos
        End If
        Set TraceDBWnd = Nothing
End Sub
```

SubItemUserWidth, TraceDBWndCmdTarget Property

Syntax SubItemUserWidth = _Integer

Description This property indicates the size in pixels of the column in the Trace display window. The -1 value is returned when the column is not displayed.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
```

```
        Dim TraceDBWnd As TraceDBWndCmdTarget
        Set TraceDBWnd = GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
        If Not TraceDBWnd Is Nothing Then
                TraceDBWnd .SubItemUserWidth = 20
                TraceDBWnd .RecalcLayout
        End If
        Set TraceDBWnd = Nothing
End Sub
```

SubItemValue, TraceDBWndCmdTarget Property

Syntax SubItemValue = _String

Description Allows you to set the text which is to appear as the "Value" column's name. The default text will be used if this field is left blank.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim TraceDBWnd As TraceDBWndCmdTarget
        Set TraceDBWnd = GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
        If Not TraceDBWnd Is Nothing Then
                TraceDBWnd .SubItemValue = "Var. Value"
                TraceDBWnd .RecalcLayout
        End If
        Set TraceDBWnd = Nothing
End Sub
```

SubItemValuePos, TraceDBWndCmdTarget Property

Syntax SubItemValuePos = _Integer

Description This property sets or returns the position of the "Value" column within the Trace DB window. When setting a new value, the other columns will be automatically re-positioned in the window layout. In addition when setting the "-1", the column will be hidden. The "0" value is used to indicate position of the first column on the left in the window.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
        Dim TraceDBWnd As TraceDBWndCmdTarget
        Set TraceDBWnd = GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
        If Not TraceDBWnd Is Nothing Then
                Debug.Print TraceDBWnd .SubItemValuePos
        End If
        Set TraceDBWnd = Nothing
End Sub
```

SubItemValueWidth, TraceDBWndCmdTarget Property

Syntax SubItemValueWidth = _Integer

Description This property indicates the size in pixels of the column in the Trace display window. The -1 value is returned when the column is not displayed.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
        Dim TraceDBWnd As TraceDBWndCmdTarget
        Set TraceDBWnd = GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
        If Not TraceDBWnd Is Nothing Then
                TraceDBWnd .SubItemValueWidth = 20
                TraceDBWnd .RecalcLayout
        End If
        Set TraceDBWnd = Nothing
End Sub
```

SubItemVarDesc, TraceDBWndCmdTarget Property

Syntax SubItemVarDesc = _String

Description Here you can enter text to show as the name for the "Variable Description" Column. The default field will be used if this field is left empty.

Parameter	Description

None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
	Dim TraceDBWnd As TraceDBWndCmdTarget
	Set TraceDBWnd =
	GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
	If Not TraceDBWnd Is Nothing Then
		TraceDBWnd.SubItemVarDesc= "Tag Description"
		TraceDBWnd.RecalcLayout
	End If
	Set TraceDBWnd = Nothing
End Sub
```

SubItemVarDescPos, TraceDBWndCmdTarget Property

Syntax SubItemVarDescPos = _Integer

Description This property sets or returns the position of the "Variable Description" column within the TraceDB Window. When setting a new value the other columns will automatically reposition within the window layout. In addition, when setting value to "-1" , the column will be hidden. the "0" value indicates the position of the first column to the left of the window.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
	Dim TraceDBWnd As TraceDBWndCmdTarget
	Set TraceDBWnd =
	GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
	If Not TraceDBWnd Is Nothing Then
		Debug.Print TraceDBWnd.SubItemVarDescPos
	End If
	Set TraceDBWnd = Nothing
End Sub
```

SubItemVarDescWidth, TraceDBWndCmdTarget Property

Syntax SubItemVarDescWidth = _Integer

Description This property indicates the size of the "Variable Description" column in pixels within the Trace window. The -1 value will return if this column is not displayed in window.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
        Dim TraceDBWnd As TraceDBWndCmdTarget
        Set TraceDBWnd =
        GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
        If Not TraceDBWnd Is Nothing Then
                TraceDBWnd.SubItemVarDescWidth = 20
                TraceDBWnd.RecalcLayout
        End If
        Set TraceDBWnd = Nothing
End Sub
```

SubItemVarGroup, TraceDBWndCmdTarget Property

Syntax SubItemVarGroup = _String

Description Here you can enter text to show as the name for the "Variable Group" Column. The default field will be used if this field is left empty.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim TraceDBWnd As TraceDBWndCmdTarget
        Set TraceDBWnd =
        GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
        If Not TraceDBWnd Is Nothing Then
                TraceDBWnd.SubItemVarGroup= "Tag Group"
                TraceDBWnd.RecalcLayout
        End If
        Set TraceDBWnd = Nothing
End Sub
```

SubItemVarGroupPos, TraceDBWndCmdTarget Property

Syntax SubItemVarGroupPos = _Integer

Description This property sets or returns the position of the "Variable Group" column within the TraceDB Window. when setting a new value the other columns will automatically reposition within the window layout. In addition, when setting value to "-1" , the column will be hidden. the "0" value indicates the position of the first column to the left of the window.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
        Dim TraceDBWnd As TraceDBWndCmdTarget
        Set TraceDBWnd =
        GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
        If Not TraceDBWnd Is Nothing Then
            Debug.Print TraceDBWnd.SubItemVarGroupPos
        End If
        Set TraceDBWnd = Nothing
End Sub
```

SubItemVarGroupWidth, TraceDBWndCmdTarget Property

Syntax SubItemVarGroupWidth = _Integer

Description This property indicates the size of the "Variable Group" column in pixels within the Trace window. The -1 value will return if this column is not displayed in window.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
        Dim TraceDBWnd As TraceDBWndCmdTarget
        Set TraceDBWnd =
        GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
        If Not TraceDBWnd Is Nothing Then
            TraceDBWnd.SubItemVarGroupWidth = 20
            TraceDBWnd.RecalcLayout
        End If
```

```
        Set TraceDBWnd = Nothing
End Sub
```

SubItemVarName, TraceDBWndCmdTarget Property

Syntax SubItemVarName = _String

Description Here you can enter text to show as the name for the "Variable Name" Column. The default field will be used if this field is left empty.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim TraceDBWnd As TraceDBWndCmdTarget
        Set TraceDBWnd = GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
        If Not TraceDBWnd Is Nothing Then
                TraceDBWnd.SubItemVarName = "Tag Name"
                TraceDBWnd.RecalcLayout
        End If
        Set TraceDBWnd = Nothing
End Sub
```

SubItemVarNamePos, TraceDBWndCmdTarget Property

Syntax SubItemVarNamePos = _Integer

Description This property sets or returns the position of the "Variable Name" column within the TraceDB Window. when setting a new value the other columns will automatically reposition within the window layout. In addition, when setting value to "-1" , the column will be hidden. the "0" value indicates the position of the first column to the left of the window.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
```

```
        Dim TraceDBWnd As TraceDBWndCmdTarget
        Set                                TraceDBWnd                                =
        GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
        If Not TraceDBWnd Is Nothing Then
                Debug.Print TraceDBWnd.SubItemVarNamePos
        End If
        Set TraceDBWnd = Nothing
End Sub
```

SubItemVarNameWidth, TraceDBWndCmdTarget Property

Syntax SubItemVarNameWidth = _Integer

Description This property indicates the size of the "Variable Name" column in pixels within the Trace window. The -1 value will return if this column is not displayed in window.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
        Dim TraceDBWnd As TraceDBWndCmdTarget
        Set                                TraceDBWnd                                =
        GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
        If Not TraceDBWnd Is Nothing Then
                TraceDBWnd.SubItemVarNameWidth = 20
                TraceDBWnd.RecalcLayout
        End If
        Set TraceDBWnd = Nothing
End Sub
```

Variable, TraceDBWndCmdTarget Property

Syntax Variable = _String

Description This property sets or returns the name of the trace table to be displayed. This method is very handy when the name of the trace table is the same as the variable's.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim TraceDBWnd As TraceDBWndCmdTarget
        Set TraceDBWnd = GetSynopticObject.GetSubObject("TraceDB").GetObjectInterface
        If Not TraceDBWnd Is Nothing Then
                Debug.Print TraceDBWnd .Variable
        End If
        Set TraceDBWnd = Nothing
End Sub
```

54. TrendCmdTarget

OnChangingState, TrendCmdTarget Event

Description Event occurs each time the trend switches over from Run to Stop and viceversa. The bSet boolean variable enables the status change, meaning that if the variable is set to False, the status change is inhibited.

Parameter	Description
bRet As Boolean	Enable on status change.

OnCursorPosChanged, TrendCmdTarget Event

Description Event occurs each time the cursor's position changes whe the tren is in 'Stop' mode. The nPos variable returns the cursor's postion in the trend's value recordset (being the number of samplings).

Parameter	Description
nPos As Long	Cursor's position in the recordset.

OnErrorRecordset, TrendCmdTarget Event

Description Event occurs when an error is verified during the creation of the recordset with the Trend values. One reason for such error could be that the query syntax is associated to the wrong trend.

This event automatically disables when the Trend or Datalogger's "Load Data in Separate Thread' execution property is enabled.

Parameter	Description
RecordsetError As String	Error message.

OnExpand, TrendCmdTarget Event

Description Event notification each time the Trend/Data Analysis area is expanded or viceversa on command.

Parameter	Description

bRet As Boolean	Set at "false" consents operation cancellation.

Example:

```
Public Sub OnExpand(ByRef bRet As Boolean)
        If MsgBox ("Do you want Expand Trend Area?", vbYesNo + vbQuestion,
        GetProjectTitle) = vbYes Then
                Debug.Print "Exapnding Trend Area..."
        Else
                Debug.Print "Deleting Expand Trend Area..."
                bRet = False
        End If
End Sub
```

OnExportEnd, TrendCmdTarget Event

Description Event occurs at the end of the exporting of values selected with the appropriate command.

Parameter	Description
None	None

OnExportNext, TrendCmdTarget Event

Description Event occurs each time the Next record of values selected with the appropriate command is exported.

Parameter	Description
NumRecord As Long	Number of records on export
bRet As Boolean	Enable continue on to next export.

OnExportStart, TrendCmdTarget Event

Description Event occurs at the beginning of the exporting of values selected with the appropriate command.

Parameter	Description
None	None

OnFailedCreatingThread, TrendCmdTarget Event

Description Event occurs when the system fails to allocate the resources for creating the thread for the trend's execution. A serious anomaly condition will be signalled where the system resources are below the minimum limit.

Parameter	Description
None	None

OnImportEnd, TrendCmdTarget Event

Description Event occurs at the end of the importing of values selected with the appropriate command.

This event automatically disables when the Trend or Datalogger's "Load Data in Separate Thread' execution property is enabled.

Parameter	Description
None	None

OnImportNext, TrendCmdTarget Event

Description Event occurs each time the next record of values, selected with the appropriate command, is imported.

This event automatically disables when the Trend or Datalogger's "Load Data in Separate Thread' execution property is enabled.

Parameter	Description
NumRecord As Long	Number of records to be imported.
bRet As Boolean	Enable to continue import procedure

OnImportStart, TrendCmdTarget Event

Description Event occurs at the start of importing values selected with the appropriate command.

This event automatically disables when the Trend or Datalogger's "Load Data in Separate Thread' execution property is enabled.

Parameter	Description
None	None

OnNext, TrendCmdTarget Event

Description Event notified each time graphics are moved to the left on command (">" button).

Parameter	Description
bRet As Boolean	Set at "False" consents operation cancellation.

Example:

```
Public Sub OnNext(ByRef bRet As Boolean)
        If MsgBox ("Do you want execute Next Command ?", vbYesNo + vbQuestion,
        GetProjectTitle) = vbYes Then
                Debug.Print "Executing Next Command..."
        Else
                Debug.Print "Deleting Next Command..."
                bRet = False
        End If
End Sub
```

OnPageChanged, TrendCmdTarget Event

Description Event occurs when a page is changed while being scrolled with the trend in pause mode.

Parameter	Description
None	None

OnPageEnd, TrendCmdTarget Event

Description Event occurs each time the last buffer's page of values has been reached while being scrolled with the trend in pause mode.

Parameter	Description
None	None

OnPageNext, TrendCmdTarget Event

Description Event notified each time the trend page is moved to the left on command (">>" button).

Parameter	Description
bRet As Boolean	Set at "false" consents operation cancellation.

Example:

```
Public Sub OnPageNext(ByRef bRet As Boolean)
        If MsgBox ("Do you want execute Page Next Command ?", vbYesNo +
        vbQuestion, GetProjectTitle) = vbYes Then
                Debug.Print "Executing Page Next Command..."
        Else
                Debug.Print "Deleting Page Next Command..."
                bRet = False
        End If
End Sub
```

OnPagePrev, TrendCmdTarget Event

Description Event notification each time a Trend page is moved to the right ("<<" button).

Parameter	Description
bRet As Boolean	Set at "false" consents operation cancellation.

Example:

```
Public Sub OnPagePrev(ByRef bRet As Boolean)
        If MsgBox ("Do you want execute Page Prev Command ?", vbYesNo +
        vbQuestion, GetProjectTitle) = vbYes Then
                Debug.Print "Executing Page Prev Command..."
        Else
                Debug.Print "Deleting Page Prev Command..."
                bRet = False
        End If
End Sub
```

OnPageStart, TrendCmdTarget Event

Description Event occurs each time the first buffer's page of values has been reached while being scrolled with the trend in pause mode.

Parameter	Description
None	None

OnPositionScrolled, TrendCmdTarget Event

Description Event occurs each time a position in the buffer of values is scrolled backwards or forwards when the trend is in pause mode.

Parameter	Description
None	None

OnPrev, TrendCmdTarget Event

Description Event notified each time the trend is moved to the right on command ("<" button).

Parameter	Description
bRet As Boolean	Set at "false" consents operation cancellation.

Example:

```
Public Sub OnPrev(ByRef bRet As Boolean)
        If MsgBox ("Do you want execute Previous Command ?", vbYesNo + vbQuestion,
        GetProjectTitle) = vbYes Then
                Debug.Print "Executing Previous Command..."
        Else
                Debug.Print "Deleting Previous Command..."
        bRet = False
        End If
End Sub
```

OnPrint, TrendCmdTarget Event

Description Event notified each time a request for printing data loaded in the Trend window has gone into effect.

This event is not supported in Window CE.

Parameter	Description
bRet As Boolean	Enabled at print start.

OnRecordsetMoveNext, TrendCmdTarget Event

Description Event occurs when the recordset of values, complying to the selection query, are scrolled each time the system acquires a new value.

This event automatically disables when the Trend or Datalogger's "Load Data in Separate Thread' execution property is enabled.

Parameter	Description
NumRecord As Long	Index of currently pointed record
bRet As Boolean	Enables continuation of scrolling procedure

OnRecordsetQueryEnd, TrendCmdTarget Event

Description Event occurs at the end of acquiring values complying to the selection query.

This event automatically disables when the Trend or Datalogger's "Load Data in Separate Thread' execution property is enabled.

Parameter	Description
None	None

OnRecordsetQueryStart, TrendCmdTarget Event

Description Event occurs at the start of the acquisition of values complying to the selection query.

This event automatically disables when the Trend or Datalogger's "Load Data in Separate Thread' execution property is enabled.

Parameter	Description
None	None

OnResetZoom, TrendCmdTarget Event

Description This event is called when existing from zoom mode, meaning that when you press the ESC key or call the "ResetZoom"method.

Please note: the "OnResetZoom" event is no longer called when existing with bRet = False from the "OnStartZoom" event. This is By-Design behaviour.

Parameter	Description
bRet As Boolean	You can exist from the Zoom mode by setting this parameter at "False".

OnStartRecording, TrendCmdTarget Event

Description Event occurs each time a trend value recording starts.

Parameter	Description
None	None

OnStartZoom, TrendCmdTarget Event

Description This event is called after the zoom area has been selected but before the zoom of the selected area is applied. Operations on data selected within the zoom area can be executed thanks to this method, for example to carry out requeries to view only the data selected.

Parameter	Description
FromRecord As Long	Position of the first value selected within the zoom area. This position is referred to the trend's buffer and other information can be retrieved with other functions ("SampleDateTime"), such as the date and time of that recording.
ToRecord As Long	Position of the last value selected in the zoom area. This position is referred to the trend's buffer and other information can be retrieved with other functions ("SampleDateTime"), such at the date and Time of that recording.
bRet As Boolean	When setting this parameter to "False" you can block the zoom of the area selected.

OnStopRecording, TrendCmdTarget Event

Description Event occurs each time the recording of the trend's values is stopped.

Parameter	Description
None	None

OnUpdateData, TrendCmdTarget Event

Description Event occurs each time an update of the trend's value on screen happens.

Parameter	Description
None	None

OnZoomAreaChanged, TrendCmdTarget Event

Description This event is called after the zoom area has been selected but before the zoom is applied to the selected area. This event returns information about the size and position of the selected zoom area.

Parameter	Description
X As Integer	Horizontal coordinate of the top left rectangle vertex representing the selected Zoom area.
Y As Integer	Vertical coordinate of the top left rectangle vertex representing the selected Zoom area.
nWidth As Integer	The rectangle's width size representing the selected zoom area.
nHeight As Integer	The rectangle's height size representing the selected zoom area.

AddPen, TrendCmdTarget Function

Syntax AddPen(lpszName, lpszVariable, dMinValue, dMaxValue, PenColor, BrushColor, nPlotType)

Description This function allows you to add a pen to the Trend object during the project run. Before executing this function you need to stop the recording procedure with the "Recording" functio.
As the Data Analysis is a tool for "Off-Line" analysis of data in the Data Base, it must be connected to a Data Logger and, unlike the Trend object, pens linked to variables cannot be added if not provided for in the Data Logger cannot

Parameter	Description
lpszName As String	Name of pen to be added
lpszVariable As String	Variable associated to pen
dMinValue As Double	Pen's minimum value
dMaxValue As Double	Pen's maximum value
PenColor As Long	Pen color

BrushColor As Long	Brush color
nPlotType As Integer	Index of line type to represent pen

Result Boolean

Example:

```
Option Explicit
Const DLColName = "SimCosInt"
Public Sub Click()
Dim strVar As String
   GetVariableNameFromList(strVar)
   If Len(strVar)<>0 Then
      objDataAnalysis.AddPen("MyPen", strVar, -100, 100, RGB(255,0,0), RGB(0,255,0),
0)
      objDataAnalysis.Refresh
      objDataAnalysis.PenDLColumnName("MyPen") = DLColName
   End If
End Sub
```

ClearAllSavedValues, TrendCmdTarget Function

Syntax ClearAllSavedValues()

Description Cleans the trend area of all the previously saved curve values. The pen values can be saved in the trend area with the SaveAllCurrentValue function and are visible in the in trend with hatched lines until the ClearAllSavedValues function is called.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        If objTrend Is Nothing Then Exit Sub
        objTrend.ClearAllSavedValues
        objTrend.Refresh
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

ClearSavedValues, TrendCmdTarget Function

Syntax ClearSavedValues(_lpszPenName)

Description Cleans the trend area of all the previously saved values of the curve specified with the lspzPenName parameter. Pen values can be saved in the trend area with the SaveCurrentValue function, and are visible with hatched lines until the ClearSavedValues function is called

Parameter	Description
lpszPenName As String	Name of the pen whose values are to be saved.

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        If objTrend Is Nothing Then Exit Sub
        objTrend.ClearSavedValues(objTrend.GetPenNameFromList(0))
        objTrend.Refresh
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

CloseBackupLink, TrendCmdTarget Function

Syntax CloseBackupLink()

Description This function closes the backup database link set with the BackupLink property and goes back to retrieve data from the original Data Logger set in the Trend. The data refresh must be forced with the Requery and Refresh functions or by changing the trend's status.

This property is not supported in Windows CE.

Parameter	Description
None	None

Result None

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        If objTrend Is Nothing Then Exit Sub
        objTrend.CloseBackupLink
        objTrend.Requery
```

```
        objTrend.Refresh
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

CopyLegendToClipboard, TrendCmdTarget Function

Syntax CopyLegendToClipboard()

Description This function copies the fields and values in the trend's legend to the Windows Clipboard in text format. Therefore the trend's values are at hand to be instantaneously returned to other files using the Windows 'Paste' command.

This property is not supported in Windows CE. (If set, always returns 'false')

This function is not managed by the "Data Analysis" object.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        If objTrend Is Nothing Then Exit Sub
        objTrend.CopyLegendToClipboard
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

EditPenProperties, TrendCmdTarget Function

Syntax EditPenProperties(_lpszPenName)

Description This function allows you to edit the pen's general settings during project run mode. The configuration window of that pen will be displayed.

This function is not supported in Windows CE.(If set, always returns 'false')

Parameter	Description

lpszPenName As String	Name of the pen to be edited.

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        If objTrend Is Nothing Then Exit Sub
        objTrend.EditPenProperties(objTrend.GetPenNameFromList(0))
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

ExportToClipboard, TrendCmdTarget Function

Syntax ExportToClipboard(_lStart, _lEnd)

Description This function executes the exporting of the specified records to the Windows Clipboard. Accepts two Long type parameters containing the indexes of the first and last record to be executed. When these parameters obtain the 0 and 0 values, the selection of records to be exported is enabled by using the mouse; in this way you can select the records by dragging the mouse pointer in the trend area. This function works only when the trend is in Stop mode.

Note: The most recent records in the time order have lower indexes.

This function is not supported in Windows CE (If set, always returns 'false').

This function is NOT managed by the "Data Analysis".

Parameter	Description
lStart As Long	Number of the first record to be copied
lEnd As Long	Number of the last record to be copied

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        If objTrend Is Nothing Then Exit Sub
        objTrend.ExportToClipboard(12,154)
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

ExportToFile, TrendCmdTarget Function

Syntax ExportToFile(_lpszFileName, _lStart, _lEnd)

Description This function exports the Trend's buffer's data to "csv" text files.

This function is NOT managed by the "Data Analysis".

Parameter	Description
lpszFileName	Destination file name
lStart As Long	Number of the first record to be copied
lEnd As Long	Number of the last record to be copied

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.ExportToFile(GetDataLoggerRecipePath & "FileData.csv", 0, -1)
End Sub

Public Sub SymbolLoading()
        Set objTrend =
        GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

GetCurrentDataLoggerName, TrendCmdTarget Function

Syntax GetCurrentDataLoggerName()

Description This function returns the name of the Data Logger used for extracting data. In cases where the Data Analysis is associated with pens deriving from diverse Data Loggers, the "OnRecordsetQueryStart" and "OnRecordsetQueryEnd" functions are called for each one fo the Data Loggers and the "GetCurrentDataLoggerName" function can be used for knowing which is the current Data Logger used for loading data.

This function is NOT managed by the "Trend" object.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Dim sDataLoggerName As String

Public Sub OnRecordsetQueryStart()
        sDataLoggerName = GetCurrentDataLoggerName()
        Debug.Print "OnRecordsetQueryStart Executed: DataLogger Name = " &
        sDataLoggerName
End Sub

Public Sub OnRecordsetQueryEnd()
        sDataLoggerName = GetCurrentDataLoggerName()
        Debug.Print "OnRecordsetQueryEnd Executed: DataLogger Name = " &
        sDataLoggerName
End Sub
```

GetCursorDateTime, TrendCmdTarget Function

Syntax GetCursorDateTime()

Description This function returns, in date format, the date and time corresponding to the cursor's position in the Trend and Data Analysis. Returns a null value when Trend is run mode.

Parameter	Description
None	None

Result Date

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        If objTrend Is Nothing Then Exit Sub
        objTrend.Freezed = True
        Debug.Print objTrend.GetCursorDateTime
End Sub
Public Sub SymbolLoading()
        Set objTrend =
        GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

GetCursorDateTimeMsec, TrendCmdTarget Function

Syntax GetCursorDateTimeMsec()

Description This function returns number of milliseconds, if recorded, together with date and time corresponding to the cursor's position in the Trend and Data Analysis. Returns a null value when Trend is run mode.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        If objTrend Is Nothing Then Exit Sub
        objTrend.Freezed = True
        Debug.Print objTrend.GetCursorDateTimeMsec
End Sub
Public Sub SymbolLoading()
        Set objTrend =
        GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

GetCursorDateTimeString, TrendCmdTarget Function

Syntax GetCursorDateTimeString()

Description This function,returns in string format,the date and time corresponding to the cursor's position in the Trend and Data Analysis. Returns a null value when Trend is run mode

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        If objTrend Is Nothing Then Exit Sub
        objTrend.Freezed = True
        Debug.Print objTrend.GetCursorDateTimeString
```

```
End Sub
Public Sub SymbolLoading()
        Set objTrend =
        GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

GetCursorPenValue, TrendCmdTarget Function

Syntax GetCursorPenValue(_lpszPenName)

Description This function returns the numeric value of the pen specified in the position corresponding to the trend's cursor. Returns a null value when the trend is in run.

This function is not managed by the "Data Analysis" object.

Parameter	Description
lpszPenName As String	Pen's name

Result Double

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        If objTrend Is Nothing Then Exit Sub
        objTrend.Freezed = True
        Debug.Print objTrend.GetCursorPenValue(objTrend.GetPenNameFromList(0))
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

GetCursorPosInLegendArea, TrendCmdTarget Function

Syntax GetCursorPosInLegendArea(_pnX, _pnY)

Description This function returns the X and Y coordinates indicating the position of the mouse cursor in respect to the legend's area origin.

Parameter	Description
pnX As Integer	Cursor's X coordinate
pnY As Integer	Cursor's Y coordinate

Result Double

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
	Dim strVar As String
	Dim pnX As Integer
	Dim pnY As Integer
	If objTrend Is Nothing Then Exit Sub
	objTrend.Freezed = True
	objTrend.GetCursorPosInLegendArea(pnX, pnY)
	Debug.Print pnX & " " & pnY
End Sub
Public Sub SymbolLoading()
	Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

GetCursorPosInPenArea, TrendCmdTarget Function

Syntax GetCursorPosInPenArea(_pnX, _pnY)

Description This function returns the X and Y coordinates indicating the mouse cursor's position in respect to the pens' area of origin.

Parameter	Description
pnX As Integer	Cursor's X coordinate
pnY As Integer	Cursor's Y coordinate

Result Double

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
	Dim strVar As String
	Dim pnX As Integer
	Dim pnY As Integer
	If objTrend Is Nothing Then Exit Sub
	objTrend.Freezed = True
	objTrend.GetCursorPosInLegendArea(pnX, pnY)
	Debug.Print pnX & " " & pnY
End Sub
Public Sub SymbolLoading()
	Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

GetCursorPosInScaleArea, TrendCmdTarget Function

Syntax GetCursorPosInScaleArea(_lpszPenName, _pnX, _pnY)

Description This function returns the X and Y coordinates indicating the mouse cursor's position in respect to the specified pen's scale area of origin.

Parameter	Description
lpszPenName As String	Pen's name
pnX As Integer	Cursor's X coordinate
pnY As Integer	Cursor's Y coordinate

Result Double

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        Dim pnX As Integer
        Dim pnY As Integer
        If objTrend Is Nothing Then Exit Sub
        objTrend.Freezed = True
        objTrend.GetCursorPosInScaleArea(objTrend.GetPenNameFromList(0),pnX, pnY)
        Debug.Print pnX & " " & pnY
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

GetCursorPosInTimeArea, TrendCmdTarget Function

Syntax GetCursorPosInTimeArea(_pnX, _pnY)

Description This function returns the X and Y coordinates indicating the mouse cursor's position in respect to time's area origin.

Parameter	Description
pnX As Integer	Cursor's X coordinate
pnY As Integer	Cursor's Y coordinate

Result Double

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
	Dim strVar As String
	Dim pnX As Integer
	Dim pnY As Integer
	If objTrend Is Nothing Then Exit Sub
	objTrend.Freezed = True
	objTrend.GetCursorPosInTimeArea(pnX, pnY)
	Debug.Print pnX & " " & pnY
End Sub
Public Sub SymbolLoading()
	Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

GetCursorPosInTrendArea, TrendCmdTarget Function

Syntax GetCursorPosInTrendArea(_pnX, _pnY)

Description This function returns the X and Y coordinates indicating the mouse cursor's position in respect to the trend window area of origin.

Parameter	Description
pnX As Integer	Cursor's X coordinate
pnY As Integer	Cursor's Y coordinate

Result Double

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
	Dim strVar As String
	Dim pnX As Integer
	Dim pnY As Integer
	If objTrend Is Nothing Then Exit Sub
	objTrend.Freezed = True
	objTrend.GetCursorPosInTrendArea(pnX, pnY)
	Debug.Print pnX & " " & pnY
End Sub
Public Sub SymbolLoading()
	Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

GetDateTimeColumnName, TrendCmdTarget Function

Syntax GetDateTimeColumnName

Description This function returns a string containing the Time Column's set name in the data logger associated to the trend.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        Dim pnX As Integer
        Dim pnY As Integer
        If objTrend Is Nothing Then Exit Sub
        objTrend.Freezed = True
        Debug.Print objTrend.GetDateTimeColumnName
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

GetFirstValidDateTime, TrendCmdTarget Function

Syntax GetFirstValidDateTime

Description This function returns, in date format, the date and time corresponding to the first valid recording in the Trend and Data Analysis value buffer. When dealing with the Trend, the value is returned whether Trend is in run or pause mode.

Parameter	Description
None	None

Result Date

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        Dim pnX As Integer
        Dim pnY As Integer
        If objTrend Is Nothing Then Exit Sub
        Debug.Print objTrend.GetFirstValidDateTime
End Sub
Public Sub SymbolLoading()
        Set objTrend =
        GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

GetFirstValidDateTimeMs, TrendCmdTarget Function

Syntax GetFirstValidDateTimeMs

Description This function returns the number of millieseconds, if recorded, together with the date and time corresponding to the first valid recording in the Trend and Data Analysis value buffer. When dealing with the Trend, the value is returned whether Trend is in run or pause mode.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        Dim pnX As Integer
        Dim pnY As Integer
        If objTrend Is Nothing Then Exit Sub
        Debug.Print objTrend.GetFirstValidDateTimeMs
End Sub
Public Sub SymbolLoading()
        Set objTrend =
        GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

GetLastValidDateTime, TrendCmdTarget Function

Syntax GetLastValidDateTime

Description This function returns, in Date format, the date and time corresponding to the last recording, therefore the most recent, in the Trend and Data Analysis value buffer. When dealing with the Trend, the value is returned whether Trend is in run or pause mode.

Parameter	Description
None	None

Result Date

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        Dim pnX As Integer
        Dim pnY As Integer
        If objTrend Is Nothing Then Exit Sub
        Debug.Print objTrend.GetLastValidDateTime
End Sub
Public Sub SymbolLoading()
        Set objTrend =
        GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

GetLastValidDateTimeMs, TrendCmdTarget Function

Syntax GetLastValidDateTimeMs

Description This function returns the number of milliseconds, if recorded, together with the date and time corresponding to the last recording, therefore the most recent, in the Trend and Data Analysis value buffer. When dealing with the Trend, the value is returned whether Trend is in run or pause mode.

Parameter	Description
None	None

Result Long

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        Dim pnX As Integer
        Dim pnY As Integer
        If objTrend Is Nothing Then Exit Sub
        Debug.Print objTrend.GetLastValidDateTimeMs
End Sub
Public Sub SymbolLoading()
        Set objTrend =
        GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

GetLastValidValuePosition, TrendCmdTarget Function

Syntax GetLastValidValuePosition

Description This function returns the value of the cursor in the position corresponding to the last recording, therefore the most recent one, in the buffer of the

Trend's values, whether in run or pause mode.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        Dim pnX As Integer
        Dim pnY As Integer
        If objTrend Is Nothing Then Exit Sub
        Debug.Print objTrend.GetLastValidValuePosition
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

GetMaxPage, TrendCmdTarget Function

Syntax GetMaxPage

Description This function returns the total number of pages displayed in the trend window based on the maximum number of samplings displayed on each page.

Parameter	Description
None	None

Result Long

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        Dim pnX As Integer
        Dim pnY As Integer
        If objTrend Is Nothing Then Exit Sub
        Debug.Print objTrend.GetMaxPage
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

GetPenNameFromList, TrendCmdTarget Function

Syntax GetPenNameFromList(_nIndex)

Description This function returns a string with the name of the pen referred to the 'nIndex' integer parameter (the first pen corresponds to the 0 index). This function will return an empty string when the index passed as parameter does not refer to any pen.

Parameter	Description
nIndex As Integer	Index the pen makes reference to.

Result String

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        Dim pnX As Integer
        Dim pnY As Integer
        If objTrend Is Nothing Then Exit Sub
        Debug.Print objTrend.GetPenNameFromList(0)
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

GetPensNumber, TrendCmdTarget Function

Syntax GetPensNumber

Description This function returns a integer type value which indicates the number of pens configured in the trend.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        Dim pnX As Integer
        Dim pnY As Integer
        If objTrend Is Nothing Then Exit Sub
        Debug.Print objTrend.GetPensNumber
```

```
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

GetTimeFontOrientation, TrendCmdTarget Function

Syntax GetTimeFontOrientation

Description This function returns a integer type value which indicates the font orientation in the trend's time area.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        Dim pnX As Integer
        Dim pnY As Integer
        If objTrend Is Nothing Then Exit Sub
        Debug.Print objTrend.GetTimeFontOrientation
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

GetZoomAreaDateTimeFrom, TrendCmdTarget Function

Syntax GetZoomAreaDateTimeFrom()

Description This function returns, in Date format, the date and time corresponding to the first point in the applied zoom area.

Parameter	Description
None	None

Result DATE

Example:

```
Option Explicit
```

```
Dim objTrend As TrendCmdTarget
Dim sZoomAreaDateTimeFrom As Date
Public Sub Click()
        Set objTrend = GetSynopticObject.GetSubObject("objTrend").GetObjectInterface
        sZoomAreaDateTimeFrom = objTrend.GetZoomAreaDateTimeFrom
        MsgBox "GetZoomAreaDateTimeFrom = " & Format(sZoomAreaDateTimeFrom,
        "YYYY/MM/DD hh:nn:ss"),vbInformation,GetProjectTitle
        Set objTrend = Nothing
End Sub
```

GetZoomAreaDateTimeMsecFrom, TrendCmdTarget Function

Syntax GetZoomAreaDateTimeMsecFrom()

Description This function returns, in numeric format, the number of milliseconds of the date and time corresponding to the first point in the applied Zoom area.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Dim sZoomAreaDateTimeMSecFrom As Integer
Public Sub Click()
        Set objTrend = GetSynopticObject.GetSubObject("objTrend").GetObjectInterface
        sZoomAreaDateTimeMSecFrom = objTrend.GetZoomAreaDateTimeMsecFrom
        MsgBox "GetZoomAreaDateTimeMsecFrom = " &
        CStr(sZoomAreaDateTimeMSecFrom),vbInformation,GetProjectTitle
        Set objTrend = Nothing
End Sub
```

GetZoomAreaDateTimeMsecTo, TrendCmdTarget Function

Syntax GetZoomAreaDateTimeMsecTo()

Description This function returns, in numeric format, the number of milliseconds of the date and time corresponding to the last point in the applied Zoom area.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Dim sZoomAreaDateTimeMSecFrom As Integer
Public Sub Click()
        Set objTrend = GetSynopticObject.GetSubObject("objTrend").GetObjectInterface
        sZoomAreaDateTimeMSecFrom = objTrend.GetZoomAreaDateTimeMsecTo
        MsgBox "GetZoomAreaDateTimeMsecTo= " & CStr(sZoomAreaDateTimeMSecFrom),vbInformation,GetProjectTitle
        Set objTrend = Nothing
End Sub
```

GetZoomAreaDateTimeStringFrom, TrendCmdTarget Function

Syntax GetZoomAreaDateTimeStringFrom()

Description This function returns, in String format, the date and time corresponding to the first point in the applied Zoom area.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Dim sZoomAreaDateTimeFrom As String
Public Sub Click()
        Set objTrend = GetSynopticObject.GetSubObject("objTrend").GetObjectInterface
        sZoomAreaDateTimeFrom = objTrend.GetZoomAreaDateTimeStringFrom
        MsgBox "GetZoomAreaDateTimeStringFrom = " & sZoomAreaDateTimeFrom,vbInformation,GetProjectTitle
        Set objTrend = Nothing
End Sub
```

GetZoomAreaDateTimeStringTo, TrendCmdTarget Function

Syntax GetZoomAreaDateTimeStringTo()

Description This function returns, in String format, the date and time corresponding to the last point in the applied Zoom area.

Parameter	Description

None	None

Result String

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Dim sZoomAreaDateTimeFrom As String
Public Sub Click()
        Set objTrend = GetSynopticObject.GetSubObject("objTrend").GetObjectInterface
        sZoomAreaDateTimeFrom = objTrend.GetZoomAreaDateTimeStringTo
        MsgBox "GetZoomAreaDateTimeStringTo = " & sZoomAreaDateTimeFrom,vbInformation,GetProjectTitle
        Set objTrend = Nothing
End Sub
```

GetZoomAreaDateTimeTo, TrendCmdTarget Function

Syntax GetZoomAreaDateTimeTo()

Description This function returns, in Date format, the date and time corresponding to the last point in the applied Zoom area.

Parameter	Description
None	None

Result DATE

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Dim sZoomAreaDateTimeFrom As Date
Public Sub Click()
        Set objTrend = GetSynopticObject.GetSubObject("objTrend").GetObjectInterface
        sZoomAreaDateTimeFrom = objTrend.GetZoomAreaDateTimeTo
        MsgBox "GetZoomAreaDateTimeTo = " & Format(sZoomAreaDateTimeFrom, "YYYY/MM/DD hh:nn:ss"),vbInformation,GetProjectTitle
        Set objTrend = Nothing
End Sub
```

GetZoomAreaScaleValueFrom, TrendCmdTarget Function

Syntax GetZoomAreaScaleValueFrom(_lpszPenName)

Description This function returns, in numeric format, the minimum scale value of the indicated pen, corresponding to the applied Zoom area.

Parameter	Description
lpszPenName as string	Name of the pen for which the minimum scale value must be recovered.

Result Integer

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Dim sZoomAreaScaleValueFrom As Double
Public Sub Click()
	Set objTrend = GetSynopticObject.GetSubObject("objTrend").GetObjectInterface
	sZoomAreaScaleValueFrom = objTrend.GetZoomAreaScaleValueFrom("Var_SignWord")
	MsgBox "GetZoomAreaScaleValueFrom = " & CStr(sZoomAreaScaleValueFrom),vbInformation,GetProjectTitle
	Set objTrend = Nothing
End Sub
```

GetZoomAreaScaleValueTo, TrendCmdTarget Function

Syntax GetZoomAreaScaleValueTo(_lpszPenName)

Description This function returns, in numeric format, the maximum scale value of the indicated pen, corresponding to the applied Zoom area.

Parameter	Description
lpszPenName as string	Name of the pen for which the maximum scale value must be recovered

Result Integer

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Dim sZoomAreaScaleValueFrom As Double
Public Sub Click()
	Set objTrend = GetSynopticObject.GetSubObject("objTrend").GetObjectInterface
	sZoomAreaScaleValueFrom = objTrend.GetZoomAreaScaleValueTo("Var_SignWord")
	MsgBox "GetZoomAreaScaleValueTo = " & CStr(sZoomAreaScaleValueFrom),vbInformation,GetProjectTitle
	Set objTrend = Nothing
End Sub
```

ImportFromClipboard, TrendCmdTarget Function

Syntax ImportFromClipboard(_lStart, _lEnd)

Description This function executes the importing of the values of the specified records from the Windows Clipboard. Aceepts two Long type parameters containing the indexes of the first and last record. When these parameters obtain the 0 and -1 respectively, all the records sized in the trend's buffer will be imported. If, however, these parameters obtain the 0 and o values, the selection of the records to be imported is enabled by using the mouse; this is done by dragging the mouse pointer in the trend area. This function only works when the trend is in Stop mode. Once this function has been called you will need to execute a trend refresh by using the appropriate command. In addition to this, all the index records higher than the last one imported will be eliminated from the trend's buffer.
All these operations have influence only on the trend's temporary values and do not influence any Data Logger that may be associated.

Tip: The most recent records in time order have the lowest indexes.

This function is not supported in Windows CE.(If set always returns 'false')

This function is not managed by the "Data Analysis" object.

Parameter	Description
lStart As Long	Number of the first record to be copied.
lEnd As Long	Number of the last record to be copied

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        If objTrend Is Nothing Then Exit Sub
        objTrend.ImportFromClipboard(12,154)
        objTrend.Refresh
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

ImportFromFile, TrendCmdTarget Function

Syntax ImportFromFile (lpszFileName, lStart, lEnd)

Description This function executes the importing of the specified record's values from a text file. Accepts a Sting type parameter containing the name of the source file and two Long type parameters containing the indexes of the first and last record to be retrieved. If these parameters obtain the 0 and

-1 values respectively, all the records sized in the trend's buffer will be imported. If, however, these parameters both obtain the 0 value, the selection of records to be exported is enabled by using the mouse; this is done by dragging the mouse pointer in the trend area. This function works only when the trend is in Stop mode. Once this function has been called you will need to execute a trend refresh by using the appropriate command. You must also keep in mind that in addition to this all the index records higher than the last imported one will be eliminated from the trend's buffer.
All these operations have effect only on the trend's temporary values and do not influence any Data Logger that may be associated.

Tip: The most recent records in time order have the lowest indexes.

This function is not managed by the "Data Analysis" object.

Parameter	Description
lpszFileName As String	Name of file to be imported
lStart As Long	Number of the first record to be copied
lEnd As Long	Number of the last record to be copied.

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        If objTrend Is Nothing Then Exit Sub
        objTrend.ImportFromFile ("exp.txt ",12,154)
        objTrend.Refresh
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

IsCursorPosInLegendArea, TrendCmdTarget Function

Syntax IsCursorPosInLegendArea

Description This function returns the True boolean value if the mouse cursor is positioned in the trend's legend area.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        If objTrend Is Nothing Then Exit Sub
        Debug.Print objTrend.IsCursorPosInLegendArea
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

IsCursorPosInPenArea, TrendCmdTarget Function

Syntax IsCursorPosInPenArea

Description This function returns the True boolen value when the mouse cursor is positioned in the trend's pen area.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        If objTrend Is Nothing Then Exit Sub
        Debug.Print objTrend.IsCursorPosInPenArea
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

IsCursorPosInScaleArea, TrendCmdTarget Function

Syntax IsCursorPosInScaleArea(_lpszPenName)

Description This function returns the True boolean value when the mouse cursor is positioned in the specified pen's scale area.

Parameter	Description
lpszPenName As String	Nome della penna

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        If objTrend Is Nothing Then Exit Sub
        Debug.Print objTrend.IsCursorPosInScaleArea(objTrend.GetPenNameFromList(0))
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

IsCursorPosInTimeArea, TrendCmdTarget Function

Syntax IsCursorPosInTimeArea

Description This function returns the True boolean value when the mouse cursor is positioned in the trend's time area.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        If objTrend Is Nothing Then Exit Sub
        Debug.Print objTrend.IsCursorPosInTimeArea
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

IsCursorPosInTrendArea, TrendCmdTarget Function

Syntax IsCursorPosInTrendArea

Description This function returns the True boolean value when the mouse cursor is positioned in the Trend window area.

Parameter	Description

None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        If objTrend Is Nothing Then Exit Sub
        Debug.Print objTrend.IsCursorPosInTrendArea
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

LinkToDataLogger, TrendCmdTarget Function

Syntax LinkToDataLogger(_bClean)

Description This function executes a trend pen update based on the Data Logger specified in the trend properties or the "LinkedDataLogger" function. The "bClean" boolean parameter allows you to specify whether to delete, or not, the pens pre-existing in the trend.

Parameter	Description
bClean As Boolean	Enables the deleting of pre-existing pens in the trend.

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        If objTrend Is Nothing Then Exit Sub
        objTrend.LinkedDataLogger = "Trend01"
        objTrend.Recording = False
        Debug.Print objTrend.LinkToDataLogger(True)
        objTrend.Recording = True
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

LoadExtSettings, TrendCmdTarget Function

Syntax LoadExtSettings

Description This function allows you to load the trend object's settings from the related file. The settings file can be specified in the properties relating to the trend or by using the "ExtSettingsFile" property. Before executing this function, the Trend must be put in pause mode.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        If objTrend Is Nothing Then Exit Sub
        objTrend.ExtSettingsFile ="Test.TSXML"
        objTrend.Recording = False
        objTrend.LoadExtSettings
        objTrend.Recording = True
End Sub
Public Sub SymbolLoading()
        Set objTrend =
        GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

MoveCursorToMousePos, TrendCmdTarget Function

Syntax MoveCursorToMousePos()

Description This function allows the Trend's cursor to be moved to the mouse pointer's position when the trend is in stop mode.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Public Sub MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
        If IsCursorPosInTrendArea Then
                MoveCursorToMousePos
        End If
End Sub
```

PageEnd, TrendCmdTarget Function

Syntax PageEnd

Description This function executes the scrolling of values displayed in the trend window until the page containing the most recent data. This function works only when the trend is in pause mode.

Parameter	Description
None	None

Result None

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        If objTrend Is Nothing Then Exit Sub
        objTrend.Freezed = True
        objTrend.PageEnd
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

PageNext, TrendCmdTarget Function

Syntax PageNext

Description This function execute the scrolling of values displayed in the trend window until the page next to the one currently displayed. The next page is the one which contains the most immediate recent data compared to the data being displayed. This function only works when the trend is in pause mode.

Parameter	Description
None	None

Result None

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        If objTrend Is Nothing Then Exit Sub
        objTrend.Freezed = True
        objTrend.PageNext
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
```

```
End Sub
```

PagePrev, TrendCmdTarget Function

Syntax PagePrev

Description This function executes the scrolling of values displayed in the trend window until the page before the one currently displayed is shown. Previous page means the one containing the most immediate oldest data in time order compared to the data being displayed. This function only works when the trend is in pause mode.

Parameter	Description
None	None

Result None

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        If objTrend Is Nothing Then Exit Sub
        objTrend.Freezed = True
        objTrend.PagePrev
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

PageStart, TrendCmdTarget Function

Syntax PageStart

Description This function executed the scrolling of values displayed in the trend window until the page containing the most oldest date in time order is shown. This function only works when the trend is in pause mode.

Parameter	Description
None	None

Result None

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        If objTrend Is Nothing Then Exit Sub
```

```
        objTrend.Freezed = True
        objTrend.PageStart
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

PrintTrend, TrendCmdTarget Function

Syntax PrintTrend(_bDefaultPrinter, _bPrintBackground, _nStartPage, _nEndPage, _lpszTitle)

Description This function permits you to print a range of trend pages. The print characteristics are those set in the Windows driver, to modify these settings in runtime you can disable the default printer by setting the first parameter (bDefualtPrinter) to False.

This function is not supported in Windows CE. To print screens use the "PrnWndCE.exe" tool.

When using the Data Analysis object, only the currently displayed page will be printed due to the fact that the Data Analysis loads the page being displayed in memory only.

Parameter	Description
bDefaultPrinter As Boolean	Uses the preset printer.
bPrintBackground As Boolean	Prints in background.
nStartPage As Integer	Number of print interval start page.
nEndPage As Integer	Number of print interval end page.
lpszTitle As String	Title to be written as printout's heading.

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        If objTrend Is Nothing Then Exit Sub
        objTrend.PrintTrend(True, True, objTrend.Page, objTrend.Page,
        objTrend.LinkedDataLogger)
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

Refresh, TrendCmdTarget Function

Syntax Refresh

Description This function executes an graphical update of the trend object. This function needs to be executed when a change has been made to a property influencing the object's graphical aspect.

Parameter	Description
None	None

Result None

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        If objTrend Is Nothing Then Exit Sub
        objTrend.Refresh
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

RemovePen, TrendCmdTarget Function

Syntax RemovePen(_lpszName)

Description This function allows you to delete a pen from the Trend when the project is being run. In order to execute this function you will need to momentarily suspend the recording by setting the Recording property to False. You may also need to use the Refresh function after removing the pen to update the trend's graphics.

Parameter	Description
lpszName As String	Name of pen

Result None

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        If objTrend Is Nothing Then Exit Sub
        objTrend.Recording = False
        objTrend.RemovePen(objTrend.GetPenNameFromList(0))
        objTrend.Recording = True
        objTrend.Refresh
End Sub
Public Sub SymbolLoading()
```

```
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

Requery, TrendCmdTarget Function

Syntax Requery

Description This function executes the edit or extraction query ("FilterBy" or "SortBy" fields"). This function must be used each time the "DataDefaultQuery" , " DataFilterBy" and "DataSortBy" properties are modified. .

Parameter	Description
None	None

Result None

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.DataFilterBy = objTrend.GetDateTimeColumnName() & " >= " & (Data -1)
        objTrend.DataSortBy = objTrend.GetDateTimeColumnName() & " DESC"
        objTrend.Requery
        objTrend.Refresh
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

ResetZoom, TrendCmdTarget Function

Syntax ResetZoom

Description This function cancels the trend's Zoom mode. Once executed, the mouse cursor, when dragging the trend area, will no longer select and zoom the area to be enlarged. .

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        If objTrend Is Nothing Then Exit Sub
        objTrend.ResetZoom
End Sub
```

```
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

RestartStatistic, TrendCmdTarget Function

Syntax RestartStatistic(_lpszPenName)

Description This function restarts the minimum, maximum and average calculation statictics for the values of the specified pen.

Parameter	Description
lpszPenName As String	Name of pen

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        If objTrend Is Nothing Then Exit Sub
        objTrend.RestartStatistic(objTrend.GetPenNameFromList(0))
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

SaveAllCurrentValue, TrendCmdTarget Function

Syntax SaveAllCurrentValue()

Description Saves all the values of the curves in the trend area. The saved values are displayed on the trend page with hatched lines until the ClearAllSavedValues function is called. This function has effect only when the ShowSavedValues property has been enabled for the pens, otherwise it will have no effect.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
```

```
        If objTrend Is Nothing Then Exit Sub
        objTrend.SaveAllCurrentValue
        objTrend.Refresh
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

SaveCurrentValue, TrendCmdTarget Function

Syntax SaveCurrentValue(_lpszPenName)

Description Saves the values of the curve specified with the lspzPenName parameter in the trend area. The saved valued are displayed on the trend page with hatched lined until the ClearSavedValues function is called. This function has effect only when the ShowSavedValues property has been enabled for the pens, otherwise it will have no effect.

Parameter	Description
lpszPenName As String	Name of the pen whose values are to be saved

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        If objTrend Is Nothing Then Exit Sub
        objTrend.SaveCurrentValue(objTrend.GetPenNameFromList(0))
        objTrend.Refresh
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

SaveExtSettings, TrendCmdTarget Function

Syntax SaveExtSettings

Description This function permits you to save the Trend object's configuration in the relating file. The configuration file can be specified in the relative Trend property or by means of using the "ExtSettingsFile" property.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.ExtSettingsFile ="Prova.TSXML"
        objTrend.SaveExtSettings
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

ScrollPosNext, TrendCmdTarget Function

Syntax ScrollPosNext(_lStep)

Description This function executes the scrolling of the trend's values, when in pause, by the number of positions specified in the direction of the most recent data.

Parameter	Description
lStep As Long	Number of positions to be scrolled

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.ScrollPosNext(4)
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

ScrollPosPrev, TrendCmdTarget Function

Syntax ScrollPosPrev(_lStep)

Description This function executed the scrolling of trend valued, when in pause, by the number of positions specified in the direction of the most oldest data.

Parameter	Description
lStep As Long	Number of positions to be scrolled

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
```

```
Public Sub Click()
        objTrend.ScrollPosNext(4)
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

SetSamplesValue, TrendCmdTarget Function

Syntax SetSamplesValue(lpszPenName, nStart, nCount, pBuffer)

Description This function allows you to force the specified pen's value of a set of consecutive samples. Accepts a String type parameter to specify the pen to be referred to, two Long type parameters to indicate the start position in the buffer and the number of samples to be modified and a Variant to pass a Double array containing the new values to be sampled.
Warning: the number of samples to be modified must be equal to the number of array elements containing the new data.

This function is not managed by the "Data Analysis" object.

Parameter	Description
lpszPenName As String	Pen's name
nStart As Long	Number of start sample
nCount As Long	Total number of samples
pBuffer As Variant	Double array containing the values

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim i As Integer
        Dim value(50) As Double

        For i = 0 To 50
        value(i) = i
        Next i

        Dim lpszPenName As String
        Dim nStart As Long
        Dim nCount As Long

        lpszPenName = "Curva1"
        nStart = 0
        nCount = 51
        objTrend.SetSamplesValue(lpszPenName, nStart, nCount, value)
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

SetTimeFontOrientation, TrendCmdTarget Function

Syntax SetTimeFontOrientation(_nNewValue)

Description This function sets the Orientation in degrees of the trend's time scale. This value starts from 0 to 45.

Parameter	Description
nNewValue As Integer	Orientation value

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim i As Integer
        Public Sub Click()
        i = i+1
        If i=5 Then i=0
        objTrend.SetTimeFontOrientation(i)
End Sub
Public Sub SymbolLoading()
Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

StartPanMode, TrendCmdTarget Function

Syntax StartPanMode

Description This function, when activated, allows you to pan the trend area by simply using the mouse. This method controls that the mouse is ready in the trend area to start panning. This function only works when the trend is in stop mode. Accepts a Boolean value.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.StartPanMode
End Sub
Public Sub SymbolLoading()
```

```
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

StartZoomMode, TrendCmdTarget Function

Syntax StartZoomMode

Description This function initializes the selection of the trend area to be displayed in zoom mode. This function can only be activated with the trend is in pause mode. When calling this function and dragging a portion of the trend area with the mouse, when the left mouse button is released, the selected area will enlarge to fill the entire trend window.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.StartZoomMode
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

AllBtnText, TrendCmdTarget Property

Syntax AllBtnText = _String

Description This property sets or returns the text to be displayed on the “All” button from the button bar for selecting time ranges in the Data Analysis object. When nothing is entered Movicon will use the default text instead.

Parameter	Description
None	None

Result String

Example:

```
Dim objDataAnalysis As TrendCmdTarget
Public Sub Click()
        If Not objDataAnalysis Is Nothing Then
                MsgBox "objDataAnalysis's AllBtnText is " &
                objDataAnalysis.AllBtnText,vbInformation,GetProjectTitle
                objDataAnalysis.AllBtnText = "All"
                objDataAnalysis.Refresh
```

```
        Else
                MsgBox "objDataAnalysis is nothing",vbExclamation,GetProjectTitle
        End If
End Sub

Public Sub SymbolLoading()
        Set objDataAnalysis =
        GetSynopticObject.GetSubObject("DataAnalysis").GetObjectInterface
End Sub
```

BackupLink, TrendCmdTarget Property

Syntax BackupLink = _String

Description This property allows you to set the ODBC connection associated to the trend. This property is useful when you need to display data from other files.

This property is not supported in Windows CE.(If set, always returns and empty string)

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetSubObject("TrendObj").GetObjectInterface
End Sub
Public Sub Click()
        Dim ChartWnd As ChartWndCmdTarget
        Dim sConnectionString As String

        sConnectionString =  "MyProject__BackupLink"  'DSN name

        If Not objTrend Is Nothing Then
                objTrend .BackupLink = sConnectionString
                objTrend .Requery
                objTrend .Refresh
        End If
        Set objTrend = Nothing
End Sub
```

BorderLegend, TrendCmdTarget Property

Syntax BorderLegend = _Boolean

Description This property enables or disables the border in the Trend's legend area.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetSubObject("TrendObj").GetObjectInterface
End Sub
Public Sub Click()
        objTrend.BorderLegend = Not objTrend.BorderLegend
        objTrend.Refresh
End Sub
```

BorderLegendRaised, TrendCmdTarget Property

Syntax BorderLegendRaised = _Boolean

Description This property sets or returns the border type for the Trend's legend area. The border is raised when set with a True boolean value.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetSubObject("TrendObj").GetObjectInterface
End Sub
Public Sub Click()
        objTrend.BorderLegendRaised = Not objTrend.BorderLegendRaised
        objTrend.Refresh
End Sub
```

BorderPen, TrendCmdTarget Property

Syntax BorderPen = _Boolean

Description This property enables or disables the border in the Trend's pen area.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetSubObject("TrendObj").GetObjectInterface
End Sub
Public Sub Click()
        objTrend.BorderPen = Not objTrend.BorderPen
        objTrend.Refresh
End Sub
```

BorderPenRaised, TrendCmdTarget Property

Syntax BorderPenRaised = _Boolean

Description This property sets or returns the border type for the trend's pen area. The border is raised when set with a True boolean value.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetSubObject("TrendObj").GetObjectInterface
End Sub
Public Sub Click()
        objTrend.BorderPenRaised = Not objTrend.BorderPenRaised
        objTrend.Refresh
End Sub
```

BorderTime, TrendCmdTarget Property

Syntax BorderTime = _Boolean

Description This property enables or disables the border in the Trend's Time area.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetSubObject("TrendObj").GetObjectInterface
End Sub
Public Sub Click()
        objTrend.BorderTime = Not objTrend.BorderTime
        objTrend.Refresh
End Sub
```

BorderTimeRaised, TrendCmdTarget Property

Syntax BorderTimeRaised = _Boolean

Description This property sets or returns the border type for the trend's time area. The border is raised when set with the True boolean value.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetSubObject("TrendObj").GetObjectInterface
End Sub
Public Sub Click()
        objTrend.BorderTimeRaised = Not objTrend.BorderTimeRaised
        objTrend.Refresh
End Sub
```

BorderTrend, TrendCmdTarget Property

Syntax BorderTrend = _Boolean

Description This property enables or disables the border in the trend's graphic area.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetSubObject("TrendObj").GetObjectInterface
End Sub
Public Sub Click()
        objTrend.BorderTrend = Not objTrend.BorderTrend
        objTrend.Refresh
End Sub
```

BorderTrendRaised, TrendCmdTarget Property

Syntax BorderTrendRaised = _Boolean

Description This property sets or returns the border type of the trend's graph area.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetSubObject("TrendObj").GetObjectInterface
End Sub
Public Sub Click()
        objTrend.BorderTrendRaised = Not objTrend.BorderTrendRaised
        objTrend.Refresh
End Sub
```

BrushColor, TrendCmdTarget Property

Syntax BrushColor = _Long

Description This property sets or returns the color associated to the brush (background area) of the indicated pen. Accepts a Long type value.

Parameter	Description
None	None

Result Long

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
	Dim strVar As String
	If objTrend Is Nothing Then Exit Sub
	objTrend.BrushColor(objTrend.GetPenNameFromList(0)) = RGB(12,56,68)
	objTrend.Refresh
End Sub
Public Sub SymbolLoading()
	Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

ButtonPos, TrendCmdTarget Property

Syntax ButtonPos = _Integer

Description This setting returns the position where the buttons are to appear in the Trend object.

The possible positions are:
0 = left
1 = top
2 = right
3 = bottom

Parameter	Description
None	None

Result Integer

Example:

```
Dim objTrend As TrendCmdTarget
Public Sub Click()
	If Not objTrend Is Nothing Then
		MsgBox "objTrend 's ButtonSize is " &
		objTrend.ButtonSize,vbInformation,GetProjectTitle
		objTrend.ButtonSize = 1
		objTrend.Refresh
	Else
		MsgBox "objTrend is nothing",vbExclamation,GetProjectTitle
	End If
End Sub

Public Sub SymbolLoading()
	Set objTrend = GetSynopticObject.GetSubObject("Trend").GetObjectInterface
End Sub
```

ButtonSize, TrendCmdTarget Property

Syntax ButtonSize = _Integer

Description This setting returns the size of the buttons which are to be displayed in the Trend object.

The possible sizes are:
0 = small
1 = medium
2 = large

Parameter	Description
None	None

Result Integer

Example:

```
Dim objTrend As TrendCmdTarget
Public Sub Click()
        If Not objTrend Is Nothing Then
                MsgBox "objTrend 's ButtonSize is " & objTrend.ButtonSize,vbInformation,GetProjectTitle
                objTrend.ButtonSize = 2
                objTrend.Refresh
        Else
                MsgBox "objTrend is nothing",vbExclamation,GetProjectTitle
        End If
End Sub

Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetSubObject("Trend").GetObjectInterface
End Sub
```

ColumnSeparator, TrendCmdTarget Property

Syntax ColumnSeparator = _Integer

Description This property sets or returns the ASCII code of the character used as the column separator in Record files associated to the trend.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        If objTrend Is Nothing Then Exit Sub
```

```
        objTrend.ColumnSeparator = Asc(",")
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

CompareTimeFrameBtnColor, TrendCmdTarget Property

Syntax CompareTimeFrameBtnColor = _Long

Description This property consents to read and set the color used for displaying buttons for the selecting Compare data time frames for the Data Analysis object. In cases where the property has been set with a new value, you will need to use the 'Refresh' method of updating the object graphically.

Parameter	Description
None	None

Result Long

Example:

```
Dim obj As TrendCmdTarget
Public Sub Click()
        Dim lColor As Long

        If obj Is Nothing Then Set obj =
        GetSynopticObject.GetSubObject("objDataAnalysis").GetObjectInterface
        If ChooseColor(lColor) Then
                obj.CompareTimeFrameBtnColor = lColor
                obj.Refresh
        End If
End Sub
```

ComposedFileName, TrendCmdTarget Property

Syntax ComposedFileName = _String

Description This property returns, in string format, the name and the path of the recording file associated to the trend. This property is read only.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        If objTrend Is Nothing Then Exit Sub
        Debug.Print objTrend.ComposedFileName
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

CompressData, TrendCmdTarget Property

Syntax CompressData = _String

Description This property sets or returns the data reading mode from the Data Logger. By setting this property with the True boolean value, the read data will be compressed to fit into one single Trend page.
When the number of records read is higher than the set number of fields to be displayed in the trend page, the trend's graphics are automatically adapted to display all the records on one or two trend pages at the most.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        If objTrend Is Nothing Then Exit Sub
        objTrend.CompressData = True
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

CurrentSelectedPen, TrendCmdTarget Property

Syntax CurrentSelectedPen = _Long

Description This property consent to read and set the pen selected in the legends. The return value corresponds to "-1" in cases where no pen has been selected in the legend. In case where the property has been set with a new value, you will need to use the "Refresh" method to update the object graphically.

Parameter	Description

None	None

Result Long

Example:

```
Dim obj As TrendCmdTarget
Public Sub Click()
	If obj Is Nothing Then Set obj =
	GetSynopticObject.GetSubObject("objTrend").GetObjectInterface
	If obj.CurrentSelectedPen >= 0 Then
		obj.EditPenProperties(obj.GetPenNameFromList(obj.CurrentSelectedPen))
		Else
		MsgBox "Plese, select a pen before!", vbInformation + vbOkOnly,
		GetProjectTitle
	End If
End Sub
```

CurrentMultiplier, TrendCmdTarget Property

Syntax CurrentMultiplier = _Byte

Description This property allows you to read or set the Multiplication factor values for the data range selection buttons.
'100' is the highest value which can be set and '1' is the lowest.
This property is only managed by the Data Analysis and not by the Trend.

Parameter	Description
None	None

Result Byte

Example:

```
Option Explicit
Dim objDataAnalysis As TrendCmdTarget
Public Sub Click()
	If Not objDataAnalysis Is Nothing Then
	objDataAnalysis.CurrentMultiplier = VAR00001
	End If
End Sub

Public Sub SymbolLoading()
	Set objDataAnalysis =
	GetSynopticObject.GetSubObject("objDataAnalysis").GetObjectInterface
End Sub
```

CurrentTopPen, TrendCmdTarget Property

Syntax CurrentTopPen = _Long

Description This property consents to read and set the first pen displayed in the legend. This consents to scrolling the pens in the legenda, when the number of pens is more than the max number of pens that can be visible at the same time (see the "MaxLegendVisiblePen" property). In cases where the property has been set with a new value, you will need to use the "Refresh" method to update the object graphically.

Parameter	Description
None	None

Result Long

Example:

```
Dim obj As TrendCmdTarget
Public Sub Click()
        If obj Is Nothing Then Set obj =
        GetSynopticObject.GetSubObject("objTrend").GetObjectInterface
                obj.CurrentTopPen = obj.CurrentTopPen + 1
        obj.Refresh
End Sub
```

CursorPos, TrendCmdTarget Property

Syntax CursorPos = _Long

Description This property sets or returns the Trend's Cursor's position in the buffer of values which are memorized when the trend is in pause mode.

Parameter	Description
None	None

Result Long

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        If objTrend Is Nothing Then Exit Sub
        objTrend.CursorPos = 100
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

DataDefaultQuery, TrendCmdTarget Property

Syntax DataDefaultQuery = _String

Description This property sets or returns the SLQ query for data extraction from the Data Logger associated to the trend object.
The "Requery" function verifies where this property is set with a value. In this case the string is sent to the ODBC and used for fetching data recordsets. The values of the other two "DataFilterBy" and "DataSortBy" properties are used only when the "DataDefaultQuery" has not been set.
This property allows custom SELECT queries to be executed. For example, the following query retrieves a recordset of a maximum of 10,000 values, where each value represents the average of values rescorded within an hour:

DataDefaultQuery = "SELECT TOP 10000 0 As MSecCol, MIN(LocalCol) As LocalCol, AVG(Cosine) As Cosine, AVG(Ramp) As Ramp FROM DataLogger GROUP BY DatePart (dayofyear, LocalCol), DatePart (Hour, LocalCol) ORDER BY LocalCol DESC"

The "DateFrom" and "DateTo" are ignored when using the "DataDefaultQuery" property for the Data Analysis.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Begin Dialog UserDialog 370,154,"TrendCmdTarget" ' %GRID:10,7,1,1
        GroupBox 20,7,340,84,"DataDefaultQuery",.GroupBox1
        TextBox 100,28,250,56,.Query,1
        Text 30,28,60,21,"Query",.Text1
        OKButton 20,105,160,42
        CancelButton 190,105,160,42
        End Dialog
        Dim dlg As UserDialog
        dlg.Query = "DELETE FROM Log1sec"
        If Dialog(dlg) <> -1 Then Exit Sub
        objTrend .DataDefaultQuery = dlg.Query
        objTrend .Requery
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

DataFileName, TrendCmdTarget Property

Syntax DataFileName = _String

Description This property sets or returns the name of the data file associated to the trend. When you want to change the name of this file you need to temporarily suspend the recording of data by setting the 'Recording' value to False. You will need to use the ".CSV" file extension otherwise the system will force this setting.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.Recording = False
        objTrend.DataFileName ="Prova"
        objTrend.Recording = True
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

DataFilterBy, TrendCmdTarget Property

Syntax DataFilterBy= _String

Description This property sets or returns the "Filter" field for extracting data from the datalogger associated to the trend object. The "FILTER" field respects the SQL syntax and corresponds to the "WHERE" clause. This clause is only used when access to the database is made, meaning at the project startup or when the trend switches over from run to stop status. When the trend is in pause mode the filter is no longer used for displaying values.
The value from the"DataFilterBy" property in the Data Analysis is used in "AND" in the WHERE clause that the Data Analysis has composed for filtering data. This consents applying the filter when the date interval buttons or scroll buttons are used or when filter is set by date.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.DataFilterBy = objTrend.GetDateTimeColumnName() & " >= " & (Data -1)
        objTrend.DataSortBy = objTrend.GetDateTimeColumnName() & " DESC"
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

DataSortBy, TrendCmdTarget Property

Syntax DataSortBy = _String

Description This property sets or returns the "SortBy" field for extracting data from the database associated to the trend object by means of the datalogger. The "SortBy" field respects the SQL syntax and corresponds to the "ORDER BY" clause. This

clause is used only when access is made to the database such as launching the project in run mode or when the trend switches over from run to stop status. When the trend is not in pause the sortby set for displaying value is not longer taken in to consideration. You need to keep in mind that the first record of the chart's values is the most recent in time order.
In the Data Analysis the "DataSortBy" property value is replaced with the one for default "TimeCol DESC".

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.DataFilterBy = objTrend.GetDateTimeColumnName() & " >= " & (Data -1)
        objTrend.DataSortBy = objTrend.GetDateTimeColumnName() & " DESC"
        objTrend.Requery
        objTrend.Refresh
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

dateFrom, TrendCmdTarget Property

Syntax dateFrom = _Date

Description This property can only be used in the "Data Analysis" object and allows you to set the start date for filtering data. It can also be sued in read for querying the last date set. In cases where the property has been set with a new value, you will need to use the "Requery" method to update and reload date from the database. When setting "dateFrom" and "dateTo" to the numeric value "0", the curve will get filters using the current date range selection for displaying data.

Parameter	Description
None	None

Result Date

Example:

```
Dim obj As TrendCmdTarget
Public Sub Click()
        If obj Is Nothing Then Set obj =
        GetSynopticObject.GetSubObject("objDataAnalysis").GetObjectInterface
        obj.dateTo = Now
        obj.dateFrom = DateAdd("n", -10, obj.dateTo)
        obj.Requery
End Sub
```

dateFromCompare, TrendCmdTarget Property

Syntax dateFromCompare = _Date

Description This property can be used only in the "Data Analysis" object and consents you to set the start date with which data is to begin filtering for the comparison curve. It can also be used in read for retrieving the last date set. In cased where the property has been set a new value, you will need to use the "Requery" method for updating and reloading data from the database. By setting "dateFromCompare" and "dateToCompare" to numeric value "0", the curve will be filtered using the current date range selection for comparing data.

Parameter	Description
None	None

Result Date

Example:

```
Dim obj As TrendCmdTarget
Public Sub Click()
        If obj Is Nothing Then Set obj =
        GetSynopticObject.GetSubObject("objDataAnalysis").GetObjectInterface
        ' Compare one minute before of the date and time on analysis
        obj.dateToCompare = DateAdd("n", -1, obj.dateTo)
        obj.dateFromCompare = DateAdd("n", -1, obj.dateFrom)
        obj.Requery
End Sub
```

DateFromCompareCurrent, TrendCmdTarget Property

Syntax dateFromCompareCurrent()

Description This property is read only and allows data to be retrieved at the beginning of the time axis used for making comparisons.

This property is managed only by the Data Analysis object and NOT the Trend object.

Parameter	Description
None	None

Result Date

Example:

```
Option Explicit
Dim objDataAnalysis As TrendCmdTarget
Public Sub SymbolLoading()
        Set objDataAnalysis = GetSynopticObject.GetSubObject
        ("objDataAnalysis").GetObjectInterface
End Sub
```

```
Public Sub Click()
        MsgBox "dateFromCompareCurrent = " & CStr
        (objDataAnalysis.dateFromCompareCurrent), vbInformation, GetProjectTitle
End Sub
```

DateFromCurrent, TrendCmdTarget Property

Syntax dateToCurrent()

Description This property is read only and consents data to be retrieved at end of time axle for the time range displayed.

This property is managed only by the Data Analysis object and NOT the Trend object.

Parameter	Description
None	None

Result Date

Example:

```
Option Explicit
Dim objDataAnalysis As TrendCmdTarget

Public Sub SymbolLoading()
        Set objDataAnalysis =
        GetSynopticObject.GetSubObject("objDataAnalysis").GetObjectInterface
End Sub

Public Sub Click()
        MsgBox "dateToCurrent = " & CStr(objDataAnalysis.dateToCurrent),
        vbInformation, GetProjectTitle
End Sub
```

dateTo, TrendCmdTarget Property

Syntax dateTo = _Date

Description This property can only be used in the "DataAnalysis" object and consents the end date to be set for finishing the filtering of data. It can also be used in read for retrieving the last data set. In cases where this property has been set with a new value, you will need to use the "Requery" method for updateding and reloading data from the database. By setting "dateFrom" and "dateTo" to the "0" numeric value, the curve will be filtered using the current date range selected for displaying data.

Parameter	Description
None	None

Result Date

Example:

```
Dim obj As TrendCmdTarget
Public Sub Click()
        If obj Is Nothing Then Set obj =
        GetSynopticObject.GetSubObject("objDataAnalysis").GetObjectInterface
        obj.dateTo = Now
        obj.dateFrom = DateAdd("n", -10, obj.dateTo)
        obj.Requery
End Sub
```

dateToCompare, TrendCmdTarget Property

Syntax dateToCompare = _Date

Description This property can only be used in the "Data Analysis" object and consents to setting the end date to stop filtering data for the comparison curve. It can also be used in read for retrieving the last date set. In cases where the property has been set with a new value, you will need to use the "Requery" method for updating and reloading data from the database. By setting "dateFromCompare" and "dateToCompare" to the numeric "0" value, the curve will be filtered using the current date range selected for displaying comparison data.

Parameter	Description
None	None

Result Date

Example:

```
Dim obj As TrendCmdTarget
Public Sub Click()
        If obj Is Nothing Then Set obj =
        GetSynopticObject.GetSubObject("objDataAnalysis").GetObjectInterface
        ' Compare one minute before of the date and time on analysis
        obj.dateToCompare = DateAdd("n", -1, obj.dateTo)
        obj.dateFromCompare = DateAdd("n", -1, obj.dateFrom)
        obj.Requery
End Sub
```

DateToCompareCurrent, TrendCmdTarget Property

Syntax dateToCompareCurrent()

Description This property is read only and consents data to be retrieved at the end of the time axis for making comparisons.

This property is only managed by the Data Analysis object and NOT the Trend object.

Parameter	Description

None	None

Result Date

Example:

```
Option Explicit
Dim objDataAnalysis As TrendCmdTarget

Public Sub SymbolLoading()
        Set objDataAnalysis = GetSynopticObject.GetSubObject
("objDataAnalysis").GetObjectInterface
End Sub

Public Sub Click()
        MsgBox "dateToCompareCurrent = " &
        CStr(objDataAnalysis.dateToCompareCurrent),
vbInformation, GetProjectTitle
End Sub
```

DateToCurrent, TrendCmdTarget Property

Syntax dateFromCurrent()

Description This property is read only and allows data to be retrieved at the beginning of the time axle for the time range displayed.

This property is only managed by the Data Analysis Object and NOT the Trend object.

Parameter	Description
None	None

Result Date

Example:

```
Option Explicit
Dim objDataAnalysis As TrendCmdTarget
Public Sub SymbolLoading()
        Set objDataAnalysis =
        GetSynopticObject.GetSubObject("objDataAnalysis").GetObjectInterface
End Sub
Public Sub Click()
        MsgBox "dateFromCurrent = " & CStr(objDataAnalysis.dateFromCurrent),
        vbInformation, GetProjectTitle
End Sub
```

DayBtnText, TrendCmdTarget Property

Syntax DayBtnText = _String

Description This property sets or returns the text to be displayed on the “Day” button from the button bar for selecting time ranges in the Data Analysis object. When nothing is entered Movicon will use the default text instead.

Parameter	Description
None	None

Result String

Example:

```
Dim objDataAnalysis As TrendCmdTarget
Public Sub Click()
        If Not objDataAnalysis Is Nothing Then
                MsgBox "objDataAnalysis's DayBtnText is " & objDataAnalysis.DayBtnText,vbInformation,GetProjectTitle
                objDataAnalysis.DayBtnText = "Day"
                objDataAnalysis.Refresh
        Else
                MsgBox "objDataAnalysis is nothing",vbExclamation,GetProjectTitle
        End If
End Sub

Public Sub SymbolLoading()
        Set objDataAnalysis = GetSynopticObject.GetSubObject("DataAnalysis").GetObjectInterface
End Sub
```

DrawGridAfter, TrendCmdTarget Property

Syntax DrawGridAfter = _Boolean

Description When enabling this property the Trend's grid comes into foreground in respect to the pens' values.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.DrawGridAfter = Not objTrend.DrawGridAfter
        objTrend.Refresh
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

ExpandBtnText, TrendCmdTarget Property

Syntax ExpandBtnText = _String

Description This property sets or returns a text for the Trend object's Expand button. If no text is entered, Movicon will use the default text instead.

Parameter	Description
None	None

Result String

Example:

```
Dim objTrend As TrendCmdTarget
Public Sub Click()
        If Not objTrend Is Nothing Then
                MsgBox "objTrend 's ExpandBtnText is " & objTrend.ExpandBtnText
                ,vbInformation,GetProjectTitle
                objTrend.ExpandBtnText = "Expand Text"
                objTrend.Refresh
        Else
                MsgBox "objTrend is nothing",vbExclamation,GetProjectTitle
        End If
End Sub

Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetSubObject("Trend").GetObjectInterface
End Sub
```

ExtSettingsFile, TrendCmdTarget Property

Syntax ExtSettingsFile = _String

Description This property sets or returns the name of the configuration file associated to the Trend. The extension for this file is ".TSXML".

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.ExtSettingsFile ="Prova.TSXML"
        objTrend.SaveExtSettings
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

FontHeightLegend, TrendCmdTarget Property

Syntax FontHeightLegend = _Integer

Description This property sets or returns the font's height used in the Trend Legend display.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.FontHeightLegend = 10
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

FontHeightScale, TrendCmdTarget Property

Syntax FontHeightScale = _Integer

Description This property sets or returns the height of the font to be used in the trend's scale.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.FontHeightScale = 10
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

FontHeightTime, TrendCmdTarget Property

Syntax FontHeightTime = _Integer

Description This property sets or returns the height of the font used in the Trend's Time display.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.FontHeightTime = 10
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

FontNameLegend, TrendCmdTarget Property

Syntax FontNameLegend = _String

Description This property sets or returns the name of the font used in the Trend's legend.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.FontNameLegend = "MS Sans Serif"
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

FontNameScale, TrendCmdTarget Property

Syntax FontNameScale = _String

Description This property sets or returns the name of the font used in the Trend's scale.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.FontNameScale = "MS Sans Serif"
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

FontNameTime, TrendCmdTarget Property

Syntax FontNameTime = _String

Description This property sets or returns the font name used in the Trend's time display.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.FontNameTime = "MS Sans Serif"
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

FormatTime, TrendCmdTarget Property

Syntax FormatTime = _String

Description This property allows you to format the date and time with which to display the dates in the Trend/Data Analysis time area. The property for displaying milliseconds remains unchanged and adds the milliseconds no matter what. The property for displaying the date however, has no effect when using this new property. When the date and hour format is customized, it will be represented on one row only. When this property is set with a new value, you will need to use the "Refresh" method to update the object graphically. All the format codes that can be used in this property have been listed in the "Time Format" property.

Parameter	Description
None	None

Result String

Example:

```
Dim obj As TrendCmdTarget
Public Sub Click()
        Dim sStyleFormat() As String
        Dim nItemSelected As Integer

        sStyleFormat = Split("%#x|%W|%a %b %Y", "|")
        nItemSelected = ShowPopupMenu(sStyleFormat)
        If nItemSelected >= 0 Then
            If obj Is Nothing Then Set obj =
            GetSynopticObject.GetSubObject("objTrend").GetObjectInterface
            obj.FormatTime = sStyleFormat(nItemSelected)
            obj.Refresh
        End If
End Sub
```

Freezed, TrendCmdTarget Property

Syntax Freezed = _Boolean

Description This property, when set at the True boolean value, freezes the Trend.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.Freezed = Not objTrend.Freezed
End Sub
```

```
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

GeneralGap, TrendCmdTarget Property

Syntax GeneralGap = _Integer

Description This property sets or returns the general gap between parts of the trend in pixels. Accepts an Integer value from 1 to 25.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetSubObject("TrendObj").GetObjectInterface
End Sub
Public Sub Click()
        objTrend.GeneralGap = 10
        objTrend.Refresh
End Sub
```

HourBtnText, TrendCmdTarget Property

Syntax HourBtnText = _String

Description This property sets or returns the text which is to be shown on the “Hour” button in the button bar for selecting time ranges in the Data Analysis object. When nothing is entered Movicon will use the default text instead.

Parameter	Description
None	None

Result String

Example:

```
Dim objDataAnalysis As TrendCmdTarget
Public Sub Click()
        If Not objDataAnalysis Is Nothing Then
                MsgBox "objDataAnalysis's HourBtnText is " &
                objDataAnalysis.HourBtnText,vbInformation,GetProjectTitle
                objDataAnalysis.HourBtnText = "Hour"
                objDataAnalysis.Refresh
        Else
```

```
                MsgBox "objDataAnalysis is nothing",vbExclamation,GetProjectTitle
        End If
End Sub

Public Sub SymbolLoading()
        Set objDataAnalysis =
        GetSynopticObject.GetSubObject("DataAnalysis").GetObjectInterface
End Sub
```

HourRecTime, TrendCmdTarget Property

Syntax HourRecTime

Description This property sets or returns the hour time frame entered in the "Record Every" property.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        Dim pnX As Integer
        Dim pnY As Integer
        If objTrend Is Nothing Then Exit Sub
        objTrend.HourRecTime = 1
        objTrend.MinRecTime = 30
        objTrend.SecRecTime = 0
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

HourViewTime, TrendCmdTarget Property

Syntax HourViewTime

Description This property sets or returns the number of hours set in the trend's View property.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        Dim pnX As Integer
        Dim pnY As Integer
        If objTrend Is Nothing Then Exit Sub
        Debug.Print objTrend.HourViewTime
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

LegendBrushColor, TrendCmdTarget Property

Syntax LegendBrushColor = _Long

Description This property set or returns the back color code of the trend's legend.

Parameter	Description
None	None

Result Long

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.LegendBrushColor = RGB(255,255,0)
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

LegendBrushVisible, TrendCmdTarget Property

Syntax LegendBrushColor = _Boolean

Description This property allows you to enable or disable the visibility of the trend's legend background.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.LegendBrushVisible = Not objTrend.LegendBrushVisible
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

LinkedDataLogger, TrendCmdTarget Property

Syntax LinkedDataLogger = _String

Description This property sets or returns the name of the Datalogger linked to the Trend object. However, once a different DataLogger has been set with this property you will need to use the "LinkToDataLogger" function to get the changes to the Trend.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Debug.Print objTrend.LinkedDataLogger
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

MaxFileLength, TrendCmdTarget Property

Syntax MaxFileLength = _Long

Description This property sets or returns the approximate length (in kb) of the text file (CSV) linkted to the Trend. The default value is 10Kb. This property has effect only when used in combination with the "StartNewFile" property or when the "Create New file" option has been enable in the Trend's settings.

Parameter	Description
None	None

Result Long

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.StartNewFile = True
        objTrend.MaxFileLength = CInt(InputBox("Max file length"))
        objTrend.SaveExtSettings
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

MaxLegendVisiblePen, TrendCmdTarget Property

Syntax MaxLegendVisiblePen = _Long

Description This property consents you to read and set the maximum number of pens visible in the legend. In cases in which the number of pens set is higher than the number of pens visible, scroll buttons will activate allowing you to scroll those within the list. When this property is set with a new value you will need to use the "Refresh" method for updating the object graphically.

Parameter	Description
None	None

Result Long

Example:

```
Dim obj As TrendCmdTarget
Public Sub Click()
        If obj Is Nothing Then Set obj =
        GetSynopticObject.GetSubObject("objTrend").GetObjectInterface
        MsgBox "obj.MaxLegendVisiblePen->" & obj.MaxLegendVisiblePen,
        vbInformation + vbOkOnly, GetProjectTitle
End Sub
```

MaxNumFiles, TrendCmdTarget Property

Syntax MaxNumFiles = _Long

Description This property sets or returns the number of text files (CSV) linked to the Trend, which will be created before being recycles. The default value is 10. This property only has effect when used in combination with the "StartNewFile" property or when the "Create New File" has been enabled in the Trend's settings.

Parameter	Description

None	None

Result Long

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.StartNewFile = True
        objTrend.MaxFileLength = CInt(InputBox("Max file length"))
        objTrend.MaxNumFiles = CInt(InputBox("Max num. files"))
        objTrend.SaveExtSettings
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

MeasureBtnText, TrendCmdTarget Property

Syntax MeasureBtnText = _String

Description This property sets or returns a text for the Data Analysis object's Measure button. When this property is set with a new value use the "Refresh" method to update the object graphically.

Parameter	Description
None	None

Result String

Example:

```
Dim obj As TrendCmdTarget
Public Sub Click()
        If obj Is Nothing Then Set obj =
        GetSynopticObject.GetSubObject("objDataAnalysis").GetObjectInterface
        If obj.MeasureBtnText <> "" Then
                obj.MeasureBtnText = ""
        Else
                obj.MeasureBtnText = "MEASURE"
        End If
        obj.Refresh
End Sub
```

MeasureTextColor, TrendCmdTarget Property

Syntax MeasureTextColor = _String

Description This property sets or returns the color assigned to the text displayed by the measure taken between a pen's points.
This property is only managed by the Data Analysis object and not the Trend.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Dim objDataAnalysis As TrendCmdTarget
Public Sub Click()

        If Not objDataAnalysis Is Nothing Then
        objDataAnalysis.MeasureTextColor = RGB(255,255,255)
        End If
End Sub

        Public Sub SymbolLoading()
        Set objDataAnalysis =
        GetSynopticObject.GetSubObject("objDataAnalysis").GetObjectInterface
End Sub
```

MinBtnText, TrendCmdTarget Property

Syntax MinBtnText = _String

Description This property sets or returns the text which is to be shown on the “Minute” button in the button bar for selecting time ranges in the Data Analysis object. When nothing is entered Movicon will use the default text instead.

Parameter	Description
None	None

Result String

Example:

```
Dim objDataAnalysis As TrendCmdTarget
Public Sub Click()
        If Not objDataAnalysis Is Nothing Then
                MsgBox "objDataAnalysis's MinBtnText is " &
                objDataAnalysis.MinBtnText,vbInformation,GetProjectTitle
                objDataAnalysis.MinBtnText= "Minutes"
                objDataAnalysis.Refresh
        Else
                MsgBox "objDataAnalysis is nothing",vbExclamation,GetProjectTitle
        End If
End Sub

Public Sub SymbolLoading()
        Set objDataAnalysis =
        GetSynopticObject.GetSubObject("DataAnalysis").GetObjectInterface
End Sub
```

MinRecTime, TrendCmdTarget Property

Syntax MinRecTime = _Integer

Description This property sets or returns the minute time frame in the "Record Every" property.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.HourRecTime = 1
        objTrend.MinRecTime = 30
        objTrend.SecRecTime = 0
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

MinViewTime, TrendCmdTarget Property

Syntax MinViewTime

Description This property sets or returns the number of minutes set in the Trend's view timeframe property.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim strVar As String
        Dim pnX As Integer
        Dim pnY As Integer
        If objTrend Is Nothing Then Exit Sub
        Debug.Print objTrend.MinViewTime
End Sub
Public Sub SymbolLoading()
```

```
        Set objTrend =
        GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

MonthBtnText, TrendCmdTarget Property

Syntax MonthBtnText = _String

Description This property sets or returns the text which is to be shown on the “Minute” button in the button bar for selecting time ranges in the Data Analysis object. When nothing is entered Movicon will use the default text instead.

Parameter	Description
None	None

Result String

Example:

```
Dim objDataAnalysis As TrendCmdTarget
Public Sub Click()
        If Not objDataAnalysis Is Nothing Then
                MsgBox "objDataAnalysis's MonthBtnText is " &
                objDataAnalysis.MonthBtnText,vbInformation,GetProjectTitle
                objDataAnalysis.MonthBtnText = "Month"
                objDataAnalysis.Refresh
        Else
                MsgBox "objDataAnalysis is nothing",vbExclamation,GetProjectTitle
        End If
End Sub

Public Sub SymbolLoading()
        Set objDataAnalysis =
        GetSynopticObject.GetSubObject("DataAnalysis").GetObjectInterface
End Sub
```

MsecRecTime, TrendCmdTarget Property

Syntax MsecRecTime = _Integer

Description This property sets or returns the number of milliseconds in the "Record Every" property.

As you can see, Movicon also allows sampling times to the millisecond, however this requires a major effect from the PC's CPU performances for which the programmer should take into full consideration.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.MsecRecTime = 30
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

NetworkBackupServerName, TrendCmdTarget Property

Syntax NetworkBackupServerName = _String

Description This property sets or returns the name of any Network Backup Server used for getting data to display in the Trend Trend or Data Analysis when the primary server, the one set in the 'NetowrkServerName'property is in timeout.

To display data from a Server, the DataLogger must also be present in the Client project, so that the Database structure can be retrieved. However, the Data Logger can only be created as structure type in the Client project, therefore without associating any variables to columns.

Parameter	Description
None	None

Result String

Example:

```
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Debug.Print objTrend.NetworkBackupServerName
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetSubObject("Trend").GetObjectInterface
End Sub
```

NetworkServerName, TrendCmdTarget Property

Syntax NetworkServerName = _String

Description This property returns the name of any Network Server where data is to be retrieved for displaying in the Trend or data Analysis.

Parameter	Description
None	None

Result String

Example:

```
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Debug.Print objTrend.NetworkServerName
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetSubObject("Trend").GetObjectInterface
End Sub
```

NextBtnText, TrendCmdTarget Property

Syntax NextBtnText = _String

Description This property sets or returns a text for the Trend object's Next command button. When nothing is entered Movicon will use the default text instead.

Parameter	Description
None	None

Result String

Example:

```
Dim objTrend As TrendCmdTarget
Public Sub Click()
        If Not objTrend Is Nothing Then
                MsgBox "objTrend 's NextBtnText is " &
                objTrend.NextBtnText,vbInformation,GetProjectTitle
                objTrend.NextBtnText = "Next"
                objTrend.Refresh
        Else
                MsgBox "objTrend is nothing",vbExclamation,GetProjectTitle
        End If
End Sub

Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetSubObject("Trend").GetObjectInterface
End Sub
```

NoneBtnText, TrendCmdTarget Property

Syntax NoneBtnText = _String

Description This property sets or returns the text which is to be shown on the “None” button in the button bar for selecting time ranges in the Data Analysis object. When nothing is entered Movicon will use the default text instead.

Parameter	Description
None	None

Result String

Example:

```
Dim objDataAnalysis As TrendCmdTarget
Public Sub Click()
	If Not objDataAnalysis Is Nothing Then
		MsgBox "objDataAnalysis's NoneBtnText is " & objDataAnalysis.NoneBtnText,vbInformation,GetProjectTitle
		objDataAnalysis.NoneBtnText = "None"
		objDataAnalysis.Refresh
	Else
		MsgBox "objDataAnalysis is nothing",vbExclamation,GetProjectTitle
	End If
End Sub

Public Sub SymbolLoading()
	Set objDataAnalysis = GetSynopticObject.GetSubObject("DataAnalysis").GetObjectInterface
End Sub
```

NumCacheRecordFile, TrendCmdTarget Property

Syntax NumCacheRecordFile = _Integer

Description This property sets or returns the number of recordings to be kept in cache memory before access to the file linked to the trend can be executed.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
	objTrend.NumCacheRecordFile = 10
End Sub
Public Sub SymbolLoading()
	Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

NumXGridDivision, TrendCmdTarget Property

Syntax NumXGridDivision = _Integer

Description This property sets or returns the maximum number of divisions for the X grid. Accepts a Integer from 1 to 10.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetSubObject("TrendObj").GetObjectInterface
End Sub
Public Sub Click()
        objTrend.NumXGridDivision = 10
        objTrend.Refresh
End Sub
```

NumXMinorGridDivision, TrendCmdTarget Property

Syntax NumXMinorGridDivision = _Integer

Description This property sets or returns the minimum number of divisions for the X grid. Accepts an integer between 1 and 10.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetSubObject("TrendObj").GetObjectInterface
End Sub
Public Sub Click()
        objTrend.NumXMinorGridDivision = 10
        objTrend.Refresh
End Sub
```

NumYGridDivision, TrendCmdTarget Property

Syntax NumYMinorGridDivision= _Integer

Description This property sets or returns the maximum number of divisions for the Y grid. Accepts an integer from 1 to 10.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetSubObject("TrendObj").GetObjectInterface
End Sub
Public Sub Click()
        objTrend.NumYGridDivision = 10
        objTrend.Refresh
End Sub
```

NumYMinorGridDivision, TrendCmdTarget Property

Syntax NumYMinorGridDivision = _Integer

Description This property sets or returns the minimum number of divisions for the Y grid. Accepts an integer from 1 to 10.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetSubObject("TrendObj").GetObjectInterface
End Sub
Public Sub Click()
        objTrend.NumYMinorGridDivision = 10
        objTrend.Refresh
End Sub
```

Page, TrendCmdTarget Property

Syntax Page = _Long

Description This property sets or returns the page number currently displayed in the trend window. Page scrolling is only allowed when the trend is in pause mode. A page represents a series of values displayed in the trend window in one unique solution. 0 index is the page containing the most recent data and the maximum limit depends on the number of sampled values and samplings per page.

Parameter	Description
None	None

Result Long

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.Freezed = True
        objTrend.Page = 5
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

PageNextBtnText, TrendCmdTarget Property

Syntax PagePrevBtnText = _String

Description This property sets or returns a text for the Trend object's Page Next command button. When nothing is specified, Movicon will use the default text instead.

Parameter	Description
None	None

Result String

Example:

```
Dim objTrend As TrendCmdTarget
Public Sub Click()
        If Not objTrend Is Nothing Then
                MsgBox "objTrend 's PagePrevBtnText is " &
                objTrend.PagePrevBtnText,vbInformation,GetProjectTitle
                objTrend.PagePrevBtnText = "Prev Page"
                objTrend.Refresh
        Else
                MsgBox "objTrend is nothing",vbExclamation,GetProjectTitle
        End If
End Sub
```

```
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetSubObject("Trend").GetObjectInterface
End Sub
```

PagePrevBtnText, TrendCmdTarget Property

Syntax PageNextBtnText = _String

Description This property sets or resets a text for the Trend object's Page Previous button. When nothing is specified, Movicon will use the default text instead.

Parameter	Description
None	None

Result String

Example:

```
Dim objTrend As TrendCmdTarget
Public Sub Click()
        If Not objTrend Is Nothing Then
                MsgBox "objTrend 's PageNextBtnText is " &
                objTrend.PageNextBtnText,vbInformation,GetProjectTitle
                objTrend.PageNextBtnText = "Next Page"
                objTrend.Refresh
        Else
                MsgBox "objTrend is nothing",vbExclamation,GetProjectTitle
        End If
End Sub

Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetSubObject("Trend").GetObjectInterface
End Sub
```

PauseRunBtnText, TrendCmdTarget Property

Syntax PauseRunBtnText = _String

Description This property sets or returns a text for the Trend object's Run/Pause button. When nothing is entered, Movicon will use the default text instead.

Parameter	Description
None	None

Result String

Example:

```
Dim objTrend As TrendCmdTarget
Public Sub Click()
```

```
        If Not objTrend Is Nothing Then
                MsgBox "objTrend 's PauseRunBtnText is " &
                objTrend.PauseRunBtnText,vbInformation,GetProjectTitle
                objTrend.PauseRunBtnText = "Run/Pause"
                objTrend.Refresh
        Else
                MsgBox "objTrend is nothing",vbExclamation,GetProjectTitle
        End If
End Sub

Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetSubObject("Trend").GetObjectInterface
End Sub
```

PenAutoscale, TrendCmdTarget Property

Syntax PenAutoscale(_lpszPenName) = _Boolean

Description This property allows you to set the pen Scale, specified in the lpszPenName parameter, in automatic mode.

Parameter	Description
lpszPenName As String	Name of pen

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.PenAutoscale(objTrend.GetPenNameFromList(0)) = Not
        objTrend.PenAutoscale(objTrend.GetPenNameFromList(0))
        objTrend.Refresh
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

PenAverageLineColor, TrendCmdTarget Property

Syntax PenAverageLineColor(_lpszPenName) = _Long

Description This property sets or returns the color of the line representing the Average values obtained by the variable linked to the pen referenced by the lpszPenName parameter.

Parameter	Description
lpszPenName As String	Name of pen

Result Long

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.PenAverageLineColor(objTrend.GetPenNameFromList(0)) = RGB(255,0,0)
        objTrend.Refresh
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

PenBackBrushPattern, TrendCmdTarget Property

Syntax PenBackBrushPattern(_lpszPenName) = _Integer

Description This property sets or returns the graphical style of the underneath the pen (property managed only when the pan is Area or Rectangle type). The different types of patterns are represented by an integer value from 0 to 5.
The default -1 value sets a solid back pattern. Values not within this range will generate a error when this function is called.

This property is not supported in Windows CE. (If set, always returns a -1)

Parameter	Description
lpszPenName As String	Pen's name

Result Integer

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.PenBackBrushPattern(objTrend.GetPenNameFromList(0))                    =
        CInt(InputBox("Insert value"))
        objTrend.Refresh
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

PenBrushColor, TrendCmdTarget Property

Syntax PenBrushColor = _Long

Description This property sets or returns the back color code of the trend's pen area.

Parameter	Description
None	None

Result Long

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.PenBrushColor  = RGB(255,0,0)
        objTrend.Refresh
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

PenBrushVisible, TrendCmdTarget Property

Syntax PenBrushVisible = _Boolean

Description This property sets or returns the Visible property of the trend's pen's area back colour.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.PenBrushVisible = Not objTrend.PenBrushVisible
        objTrend.Refresh
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

PenColor, TrendCmdTarget Property

Syntax PenColor(_lpszPenName) = _Long

Description This property sets or returns the color of the pen referenced by the lpszPenName parameter.

Parameter	Description
lpszPenName As String	Name of pen

Result Long

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.PenColor(objTrend.GetPenNameFromList(0))  = RGB(255,0,0)
        objTrend.Refresh
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

PenDLColumnName, TrendCmdTarget Property

Syntax PenDLColumnName("_lpszPenName") = _String

Description This property in read controls whether the trend is associated to a datalogger, and if this the case, returns the Data Logger Col property value for the pen passed as parameter. If the Data Logger Col has not been specified, the name of the pen is returned.
In write, it set the value of the Data Logger Col value for the pen passed as parameter.

Parameter	Description
lpszPenName As String	Trend pen name to which the datalogger column has been associated.

Result String

Example:

```
Public Sub Click()
Dim objTrend As TrendCmdTarget
Dim sPenName As String
Dim sPenDLName As String
Dim i As Integer
        Set objTrend = GetSynopticObject.GetSubObject("Trend").GetObjectInterface
        For i= 0 To objTrend.GetPensNumber()
                sPenName = objTrend.GetPenNameFromList(i)
                If sPenName <> "" Then
'get DataLogger column name
                sPenDLName = objTrend.PenDLColumnName(sPenName)
                MsgBox "PenName = " & sPenName & "; DLColName = " & sPenDLName
'set DataLogger column name
                sPenDLName = "Col" & i
                objTrend.PenDLColumnName(sPenName) = sPenDLName
                MsgBox "PenName = " & sPenName & " ; DLColName = " & PenDLName
        End If
        Next
End sub
```

PenDLRName, TrendCmdTarget Property

Syntax PenDLRName(_lpszPenName) = _String

Description This property is used for reading or setting the name of the pen's reference Data Logger. When a pen's reference Data Logger name is changed you will need to use the "LinkToDataLogger" function until changes have been applied to the Data Analysis.

This property is not managed by the "Trend" object.

Parameter	Description
lpszPenName As String	Name of the Data Analysis pen to be associated to the Data Logger.

Result String

Example:

```
Dim objDataAnalysis As TrendCmdTarget
Public Sub Click()
        Set objDataAnalysis =
        GetSynopticObject.GetSubObject("objDataAnalysis").GetObjectInterface
        objDataAnalysis.PenDLRName(sPenName) = sDataLoggerName
        objDataAnalysis.LinkToDataLogger(True)
        objDataAnalysis.Refresh
        Set objDataAnalysis = Nothing
End Sub
```

PenEditable, TrendCmdTarget Property

Syntax PenEditable(_lpszPenName) = _Boolen

Description This property enables or disables the option to edit the properties of the pen, referenced by the lpszPenName parameter, in run time.

Parameter	Description
lpszPenName As String	Name of pen

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.PenEditable(objTrend.GetPenNameFromList(0)) = Not
        objTrend.PenEditable(objTrend.GetPenNameFromList(0))
        objTrend.Refresh
```

```
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

PenFormatScale, TrendCmdTarget Property

Syntax PenFormatScale(_lpszPenName) = _String

Description This property sets the format representing the numeric values displayed in the scale of the pen referenced by the lpszPenName parameter.

Parameter	Description
lpszPenName As String	Name of pen

Result String

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.PenFormatScale(objTrend.GetPenNameFromList(0))  = = "xx.xx" 'i.e. for xx.xx -
        > value 3.7 is represented like 03.70
        objTrend.Refresh
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

PenLogarithmicScale, TrendCmdTarget Property

Syntax PenLogarithmicScale(_lpszPenName) = _Boolean

Description This property enables or disables the Logarithmic scale for the pen referenced by the lpszPenName parameter.

Parameter	Description
lpszPenName As String	Name of pen

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.PenLogarithmicScale(objTrend.GetPenNameFromList(0)) = True
        objTrend.Refresh
```

```
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

PenMaxLineColor, TrendCmdTarget Property

Syntax PenMaxLineColor(_lpszPenName) = _Long

Description This property sets or returns the color of the line representing the Maximun value obtained by the variable lined to the pen referenced by the lpszPenName parameter.

Parameter	Description
lpszPenName As String	Name of pen

Result Long

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.PenMaxLineColor(objTrend.GetPenNameFromList(0)) = RGB(255,0,0)
        objTrend.Refresh
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

PenMaxValue, TrendCmdTarget Property

Syntax PenMaxValue(_lpszPenName) = _Double

Description This property sets or returns the maximum value for the specified pen's scale. After this value has been set you will need to execute a Refresh so that this modification is shown graphically.

Parameter	Description
lpszPenName As String	Name of pen

Result Double

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.PenMaxValue(objTrend.GetPenNameFromList(0)) = 200
        objTrend.Refresh
End Sub
```

```
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

PenMinLineColor, TrendCmdTarget Property

Syntax PenMinLineColor(_lpszPenName) = _Long

Description This property sets or returns the color of the line representing the Minimum value obtained by the variable linked to the pen referenced by the lpszPenName parameter.

Parameter	Description
lpszPenName As String	Name of pen

Result Long

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.PenMinLineColor(objTrend.GetPenNameFromList(0)) = RGB(255,0,0)
        objTrend.Refresh
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

PenMinValue, TrendCmdTarget Property

Syntax PenMinValue(_lpszPenName) = _Double

Description This property sets or returns the minimum value for the specified pen's scale. After this value has been set you will need to execute a Refresh so that this modification is shown graphically.

Parameter	Description
lpszPenName As String	Name of pen

Result Double

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.PenMinValue(objTrend.GetPenNameFromList(0)) = 0
        objTrend.Refresh
End Sub
Public Sub SymbolLoading()
```

```
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

PenPlotType, TrendCmdTarget Property

Syntax PenPlotType(_lpszPenName) = _Integer

Description This property sets or returns the index relating to the line (or representation type) used for the specified pen. The ePlottingType enumerator can be used for these values:

The values are:
enum_PT_POLYLINE = line
enum_PT_POLYRECTANGLE = rectangle
enum_PT_POLYAREA = area
enum_PT_POLYLINEANDLINE = line area
enum_PT_POLYLINESTEP = line in Steps

Parameter	Description
lpszPenName As String	Name of pen

Result Double

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.PenPlotType(objTrend.GetPenNameFromList(0)) = enum_PT_POLYAREA
        objTrend.Refresh
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

PenScaleRightBottom, TrendCmdTarget Property

Syntax PenScaleRightBottom(_lpszPenName) = _Boolean

Description This property sets or returns the position of the scale relating to the referenced pen.

The value options are:
False=Top/Left
True=Bottom/Right

Parameter	Description
lpszPenName As String	Name of pen

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
	Dim ScalePos As Boolean

	ScalePos = objTrend.PenScaleRightBottom(objTrend.GetPenNameFromList(0))
	objTrend.PenScaleRightBottom(objTrend.GetPenNameFromList(0)) = Not ScalePos
	objTrend.Refresh
End Sub
Public Sub SymbolLoading()
	Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

PenShowAverageLine, TrendCmdTarget Property

Syntax PenShowAverageLine(_lpszPenName) = _Boolean

Description This property enables or disables the showing of the average value line for the specified pen.

Parameter	Description
lpszPenName As String	Name of pen

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
	Dim tmp As Boolean

	tmp = objTrend.PenShowAverageLine(objTrend.GetPenNameFromList(0))
	objTrend.PenShowAverageLine(objTrend.GetPenNameFromList(0)) = Not tmp
	objTrend.Refresh
End Sub
Public Sub SymbolLoading()
	Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

PenShowMaxLine, TrendCmdTarget Property

Syntax PenShowMaxLine(_lpszPenName) = _Boolean

Description This property enables or disables the displaying of the maximum value line for the specified pen.

Parameter	Description
lpszPenName As String	Name of pen

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim tmp As Boolean

        tmp = objTrend.PenShowMaxLine(objTrend.GetPenNameFromList(0))
        objTrend.PenShowMaxLine(objTrend.GetPenNameFromList(0)) = Not tmp
        objTrend.Refresh
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

PenShowMinLine, TrendCmdTarget Property

Syntax PenShowMinLine(_lpszPenName) = _Boolean

Description This property enables or disables the displaying of the minimum value line for the specified pen.

Parameter	Description
lpszPenName As String	Name of pen

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim tmp As Boolean

        tmp = objTrend.PenShowMinLine(objTrend.GetPenNameFromList(0))
        objTrend.PenShowMinLine(objTrend.GetPenNameFromList(0)) = Not tmp
        objTrend.Refresh
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

PenShowScale, TrendCmdTarget Property

Syntax PenShowScale(_lpszPenName) = _Boolean

Description This property enables or disables the displaying of the scale for the pen specified.

Parameter	Description
lpszPenName As String	Name of pen

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim tmp As Boolean

        tmp = objTrend.PenShowScale(objTrend.GetPenNameFromList(0))
        objTrend.PenShowScale(objTrend.GetPenNameFromList(0)) = Not tmp
        objTrend.Refresh
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

PenSize, TrendCmdTarget Property

Syntax PenSize = _Integer

Description This property sets or returns the size, in pixels, of the pen area. Accepts an integer between 5 to 25.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetSubObject("TrendObj").GetObjectInterface
End Sub
Public Sub Click()
        objTrend.PenSize = 10
        objTrend.Refresh
End Sub
```

PenSize, TrendCmdTarget Property

Syntax PenSize = _Integer

Description This property sets or returns the size, in pixels, of the pen area. Accepts an integer between 5 to 25.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetSubObject("TrendObj").GetObjectInterface
End Sub
Public Sub Click()
        objTrend.PenSize = 10
        objTrend.Refresh
End Sub
```

PenStyle, TrendCmdTarget Property

Syntax PenStyle("_lpszPenName") = _Short

Description This property sets or returns the style associated to the pen specified by the lpszPenName parameter.
The valid values are:

0 SOLID
1 DASH
2 DOT
3 DASHDOT
4 DASHDOTDOT
5 NULL

Note:
The DOT, DASHDOT and DASHDOTDOT values cannot be used in Windows CE as not supported.

Parameter	Description
lpszPenName As String	Nome della penna del trend alla quale assegnare lo stile grafico

Result Short

Example:

```
Public Sub Click()
Dim objTrend As TrendCmdTarget
Dim sPenName As String
```

```
Dim nPenStyle as Integer

        Set objTrend = GetSynopticObject.GetSubObject("Trend").GetObjectInterface
        For i= 0 To objTrend.GetPensNumber()
                sPenName = objTrend.GetPenNameFromList(i)
                If sPenName <> "" Then
'get pen style
                nPenStyle = objTrend.PenStyle(sPenName)
'set pen style
                If nPenStyle < 0 Or nPenStyle > 5 Then
                        nPenStyle = 0
                Else
                        nPenStyle = nPenStyle + 1
                End If
                objTrend.PenStyle(sPenName) = nPenStyle
        End If
        Next
End sub
```

PenVariable, TrendCmdTarget Property

Syntax PenVariable(_lpszPenName) = _String

Description This property sets or returns the name of the variable associated to the specified pen. When you wish to modify this property you need to momentarily suspend the recording by setting the Recording property to False. Any modifications done to the variable associated to the pen will not influence the variables of any linked Datalogger.

Parameter	Description
lpszPenName As String	Name of pen

Result String

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim tmp As Boolean
        Debug.Print objTrend.PenVariable(objTrend.GetPenNameFromList(0))
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

PenWidth, TrendCmdTarget Property

Syntax PenWidth(_lpszPenName) = _Integer

Description This property sets or returns the width of the specified pen. The values which you can use start from 1 to 5. An error will be generated if you use any values not within this range.

Parameter	Description
lpszPenName As String	Name of pen

Result Integer

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim tmp As Boolean
        objTrend.PenWidth(objTrend.GetPenNameFromList(0)) = 2
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

PrevBtnText, TrendCmdTarget Property

Syntax PrevBtnText = _String

Description This property sets or returns a text for the Trend's Previous command button. When nothing has been specified, Movicon will use the default text instead.

Parameter	Description
None	None

Result String

Example:

```
Dim objTrend As TrendCmdTarget
Public Sub Click()
        If Not objTrend Is Nothing Then
                MsgBox "objTrend 's PrevBtnText is " &
                objTrend.PrevBtnText,vbInformation,GetProjectTitle
                objTrend.PrevBtnText = "Prev"
                objTrend.Refresh
        Else
                MsgBox "objTrend is nothing",vbExclamation,GetProjectTitle
        End If
End Sub

Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetSubObject("Trend").GetObjectInterface
End Sub
```

PrintBtnText, TrendCmdTarget Property

Syntax PrintBtnText = _String

Description This property sets or returns a text for the Trend object's Print command button. When nothing has been specified, Movicon will use the default text instead.

This property is not supported in Windows CE.(If set, always returns an empty string)

Parameter	Description
None	None

Result String

Example:

```
Dim objTrend As TrendCmdTarget
Public Sub Click()
	If Not objTrend Is Nothing Then
		MsgBox "objTrend 's PrintBtnText is " & objTrend.PrintBtnText,vbInformation,GetProjectTitle
		objTrend.PrintBtnText = "Print"
		objTrend.Refresh
	Else
		MsgBox "objTrend is nothing",vbExclamation,GetProjectTitle
	End If
End Sub

Public Sub SymbolLoading()
	Set objTrend = GetSynopticObject.GetSubObject("Trend").GetObjectInterface
End Sub
```

Recording, TrendCmdTarget Property

Syntax Recording = _Boolean

Description This property enables or disables the recording in the trend of values associated to the pens.

Parameter	Description
lpszPenName As String	Name of pen

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
	Dim tmp As Boolean

	tmp = objTrend.Recording
	objTrend.Recording = Not tmp
	objTrend.Refresh
End Sub
Public Sub SymbolLoading()
	Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
```

```
End Sub
```

RecordOnFile, TrendCmdTarget Property

Syntax RecordOnFile = _Boolean

Description This property enables or disables the recording on file, in "CSV" format, of data sampled by the trend object. When you wish to change this setting you must momentarily suspend the recording by setting the Recording property to False.

Parameter	Description
lpszPenName As String	Name of pen.

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim tmp As Boolean

        objTrend.Recording = False
        tmp = objTrend.RecordOnFile
        objTrend.RecordOnFile = Not tmp
        objTrend.Recording = True
        objTrend.Refresh
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

SampleDateTime, TrendCmdTarget Property

Syntax SampleDateTime(_lPosition) = _Date

Description This property sets or returns the data and time, in Date format, of the sample indicated as parameter. Any date modifications influence only the value in the trend's buffer and in the file or Data Logger associated.

This function is not managed by the "Data Analysis" object.

Parameter	Description
lPosition As Long	Numero del campionamento

Result Date

Example:

```
Option Explicit
```

```
Dim objTrend As TrendCmdTarget
Public Sub Click()
        If objTrend Is Nothing Then Exit Sub
        Debug.Print objTrend.SampleDateTime(0)
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

SampleDateTimeMs, TrendCmdTarget Property

Syntax SampleDateTimeMs(_lPosition) = _Integer

Description This property sets or resets the number of milliseconds, combined with the date and time, of the sample indicated as parameter. Any data modifications will influence values in the trend's buffer and not data on file or in the associated Data Logger.

Parameter	Description
lPosition As Long	Sample number

Result Integer

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        If objTrend Is Nothing Then Exit Sub
        Debug.Print objTrend.SampleDateTimeMs(0)
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

SamplePerUpdate, TrendCmdTarget Property

Syntax SamplePerUpdate = _Integer

Description This property sets or returns the number of samples for each graphic update, this means the number of sampled data to be displayed for each trend page scroll. When you wish to change this setting you need to momentarily suspend the recording by setting the Recording property to False.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        If objTrend Is Nothing Then Exit Sub
        objTrend.Recording = False
        objTrend.SamplePerUpdate = 56
        objTrend.Recording = True
        objTrend.Refresh
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

Samples, TrendCmdTarget Property

Syntax Samples = _Long

Description This property shows the total number of samples the trend or the Data Analysis can handle. The trend has its own buffer which can be set to a limited value of 10,000 maximum. The Data Analysis has no limits.

Parameter	Description
None	None

Result Long

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim s As String
        If objTrend Is Nothing Then
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
        End If
        s$ = InputBox("VAL.:",  "Samples",CStr(objTrend.Samples))
        If s$<>"" Then
        objTrend.Recording = False
        objTrend.Samples = (CLng(s$))
        objTrend.Recording = True
        objTrend.Refresh
        End If
End Sub
```

SampleValue, TrendCmdTarget Property

Syntax SampleValue(lpszPenName, lPosition) = Double

Description This property sets or returns the sample value of the pen indicated by the position parameter in the trend's buffer.

This function is not managed by the "Data Analysis" object.

Parameter	Description
lpszPenName As String lPosition As Long	Pen's name Sample number

Result Double

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim s As String
        If objTrend Is Nothing Then
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
        End If
        Dim i As Long
        For i = 1 To 100
          Debug.Print objTrend.SampleValue(objTrend.GetPenNameFromList(0),i)
          objTrend.SampleValue(objTrend.GetPenNameFromList(0),i) = 0
        Next i
End Sub
```

ScrollPosition, TrendCmdTarget Property

Syntax ScrollPosition = _Long

Description This read only property returns the number of the most recent samples shown on the currently Trend page when the trend is in pause mode.

Parameter	Description
lpszPenName As String	Pen's name
lPosition As Long	Sample number

Result Long

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        If objTrend Is Nothing Then
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
        End If
        Debug.Print objTrend.ScrollPosition
End Sub
```

SecBtnText, TrendCmdTarget Property

Syntax SecBtnText = _String

Description This property sets or returns the text to be displayed in the scroll buttons when selecting the Data Analysis object's Minute time ranges to be displayed. When nothing has been specified, Movicon will use the default text instead.

Parameter	Description
None	None

Result String

Example:

```
Dim objDataAnalysis As TrendCmdTarget
Public Sub Click()
        If Not objDataAnalysis Is Nothing Then
                MsgBox "objDataAnalysis's SecBtnText is " &
                objDataAnalysis.SecBtnText,vbInformation,GetProjectTitle
                objDataAnalysis.SecBtnText= "Seconds"
                objDataAnalysis.Refresh
        Else
                MsgBox "objDataAnalysis is nothing",vbExclamation,GetProjectTitle
        End If
End Sub

Public Sub SymbolLoading()
        Set objDataAnalysis =
        GetSynopticObject.GetSubObject("DataAnalysis").GetObjectInterface
End Sub
```

SecRecTime, TrendCmdTarget Property

Syntax SecRecTime = _Integer

Description This property sets or returns the seconds time frame on the "Record Every" property.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim objTrend As TrendCmdTarget
        Set objTrend = GetSynopticObject.GetSubObject("Trend").GetObjectInterface
        objTrend.HourRecTime = 1
        objTrend.MinRecTime = 30
```

```
        objTrend.SecRecTime = 0
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

SecViewTime, TrendCmdTarget Property

Syntax SecViewTime = _Integer

Description This property sets or returns the number of seconds set for the View time interval.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim objTrend As TrendCmdTarget
        Set objTrend = GetSynopticObject.GetSubObject("Trend").GetObjectInterface
        objTrend.SecViewTime = 20
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

ShiftGrid, TrendCmdTarget Property

Syntax ShiftGrid = _Boolean

Description This property sets or returns the shifting option of the Trend's grid. Accepts a boolean value.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetSubObject("TrendObj").GetObjectInterface
End Sub
```

```
Public Sub Click()
        objTrend.ShiftGrid = Not objTrend.ShiftGrid
        objTrend.Refresh
End Sub
```

ShowBreakLines, TrendCmdTarget Property

Syntax ShowBreakLines = _Boolean

Description This property sets or returns the Break Lines option. The break lines are lines perpendicular to the trend's direction which indicate; when the trend is in pause mode, the points in which the recording of values was interrupted for a time higher than the set sample period.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetSubObject("TrendObj").GetObjectInterface
End Sub
Public Sub Click()
        objTrend.ShowBreakLines = Not objTrend.ShowBreakLines
        objTrend.Refresh
End Sub
```

ShowCompareTimeFrameBtn, TrendCmdTarget Property

Syntax ShowCompareTimeFrameBtn = _Boolean

Description This property permits you to display or hide the buttons for selecting the data comparison time frames for the Data Analysis object. In cases where this property has been set with a new value, you will need to use the "Refresh" method to update the object graphically.

Parameter	Description
None	None

Result Boolean

Example:

```
Dim obj As TrendCmdTarget
Public Sub Click()
```

```
        If obj Is Nothing Then Set obj =
        GetSynopticObject.GetSubObject("objDataAnalysis").GetObjectInterface
        obj.ShowCompareTimeFrameBtn = Not obj.ShowCompareTimeFrameBtn
        obj.Refresh
End Sub
```

ShowDate, TrendCmdTarget Property

Syntax ShowDate = _Boolean

Description This property sets or returns the option to enable the showing of the date. Accepts a boolean value.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetSubObject("TrendObj").GetObjectInterface
End Sub
Public Sub Click()
        objTrend.ShowDate = Not objTrend.ShowDate
        objTrend.Refresh
End Sub
```

ShowExpandBtn, TrendCmdTarget Property

Syntax ShowExpandBtn = _Boolean

Description This property permits the Trend object's Expand command to be displayed or hidden.

Parameter	Description
None	None

Result Boolean

Example:

```
Dim objTrend As TrendCmdTarget
Public Sub Click()
        If Not objTrend Is Nothing Then
                MsgBox "objTrend 's ShowExpandBtn is " &
                objTrend.ShowExpandBtn,vbInformation,GetProjectTitle
                objTrend.ShowExpandBtn = Not objTrend.ShowExpandBtn
```

```
                objTrend.Refresh
        Else
                MsgBox "objTrend is nothing",vbExclamation,GetProjectTitle
        End If
End Sub

Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetSubObject("Trend").GetObjectInterface
End Sub
```

ShowFirstPointBtn, TrendCmdTarget Property

Syntax ShowFirstPointBtn = _Boolean

Description This property makes the scroll cursor button visible/not visible which is used for moving the cursor to the first point of the curve displayed in the Data Analysis. This property is only managed by the Data Analysis and not the Trend object.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objDataAnalysis As TrendCmdTarget
Public Sub Click()

        If Not objDataAnalysis Is Nothing Then
        objDataAnalysis.ShowFirstPointBtn = Not objDataAnalysis.ShowFirstPointBtn
        End If
End Sub

Public Sub SymbolLoading()
        Set objDataAnalysis =
        GetSynopticObject.GetSubObject("objDataAnalysis").GetObjectInterface
End Sub
```

ShowLastPointBtn, TrendCmdTarget Property

Syntax ShowLastPointBtn = _Boolean

Description This property makes the scroll cursor button visible/not visible which is used for moving the cursor to the last point of the curve displayed in the Data Analysis. This property is only managed by the Data Analysis and not the Trend object.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objDataAnalysis As TrendCmdTarget
Public Sub Click()

        If Not objDataAnalysis Is Nothing Then
                objDataAnalysis.ShowLastPointBtn = Not
                objDataAnalysis.ShowLastPointBtn
        End If
End Sub

Public Sub SymbolLoading()
        Set objDataAnalysis =
        GetSynopticObject.GetSubObject("objDataAnalysis").GetObjectInterface
End Sub
```

ShowLegend, TrendCmdTarget Property

Syntax ShowLegend = _Boolean

Description This property sets or returns the option which enables or disables the showing of the legend. Accepts a boolean value.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetSubObject("TrendObj").GetObjectInterface
End Sub
Public Sub Click()
        objTrend.ShowLegend = Not objTrend.ShowLegend
        objTrend.Refresh
End Sub
```

ShowMeasureBtn, TrendCmdTarget Property

Syntax ShowMeasureBtn = _Boolean

Description This property permits you to display or hide the Measure buttons for the Data Analysis object. In cases where this property has been set with a new value, you will need to used the "Refresh" method to update the object graphically.

Parameter	Description

None	None

Result Boolean

Example:

```
Dim obj As TrendCmdTarget
Public Sub Click()
        If obj Is Nothing Then Set obj =
        GetSynopticObject.GetSubObject("objDataAnalysis").GetObjectInterface
        obj.ShowMeasureBtn = Not obj.ShowMeasureBtn
        obj.Refresh
End Sub
```

ShowMinorXGrid, TrendCmdTarget Property

Syntax ShowMinorXGrid = _Boolean

Description This property sets or returns the option for enabling or disabling the showing of the minor X grid. Accepts a boolean value.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetSubObject("TrendObj").GetObjectInterface
End Sub
Public Sub Click()
        objTrend.ShowMinorXGrid = Not objTrend.ShowMinorXGrid
        objTrend.Refresh
End Sub
```

ShowMinorYGrid, TrendCmdTarget Property

Syntax ShowMinorYGrid = _Boolean

Description This property sets or returns the option for enabling or disabling the showing of the minor Y Grid. Accepts a boolean value.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetSubObject("TrendObj").GetObjectInterface
End Sub
Public Sub Click()
        objTrend.ShowMinorYGrid = Not objTrend.ShowMinorYGrid
        objTrend.Refresh
End Sub
```

ShowMsec, TrendCmdTarget Property

Syntax ShowMsec = _Boolean

Description This property sets or returns the option for enabling or disabling the showing of milliseconds in the Trend's time. Accepts a boolean value.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetSubObject("TrendObj").GetObjectInterface
End Sub
Public Sub Click()
        objTrend.ShowMsec = Not objTrend.ShowMsec
        objTrend.Refresh
End Sub
```

ShowNextBtn, TrendCmdTarget Property

Syntax ShowNextBtn = _Boolean

Description This property permits the Trend object's Next command button to be displayed or hidden.

Parameter	Description
None	None

Result Boolean

Example:

```
Dim objTrend As TrendCmdTarget
Public Sub Click()
        If Not objTrend Is Nothing Then
                MsgBox "objTrend 's ShowNextBtn is " &
                objTrend.ShowNextBtn,vbInformation,GetProjectTitle
                objTrend.ShowNextBtn = Not objTrend.ShowNextBtn
                objTrend.Refresh
        Else
                MsgBox "objTrend is nothing",vbExclamation,GetProjectTitle
        End If
End Sub

Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetSubObject("Trend").GetObjectInterface
End Sub
```

ShowNextPointBtn, TrendCmdTarget Property

Syntax ShowNextPointBtn = _Boolean

Description This property makes the scroll cursor button visible/not visible which is used for moving the cursor to the next point of the curve displayed in the Data Analysis. This property is only managed by the Data Analysis and not the Trend object.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objDataAnalysis As TrendCmdTarget
Public Sub Click()
        If Not objDataAnalysis Is Nothing Then
                objDataAnalysis.ShowNextPointBtn= Not
                objDataAnalysis.ShowNextPointBtn
        End If
End Sub

Public Sub SymbolLoading()
        Set objDataAnalysis =
        GetSynopticObject.GetSubObject("objDataAnalysis").GetObjectInterface
End Sub
```

ShowPageNextBtn, TrendCmdTarget Property

Syntax ShowPageNextBtn = _Boolean

Description This property permits the Trend object's Page Next command button to be displayed or hidden.

Parameter	Description
None	None

Result Boolean

Example:

```
Dim objTrend As TrendCmdTarget
Public Sub Click()
	If Not objTrend Is Nothing Then
		MsgBox "objTrend 's ShowPageNextBtn is " & objTrend.ShowPageNextBtn,vbInformation,GetProjectTitle
		objTrend.ShowPageNextBtn = Not objTrend.ShowPageNextBtn
		objTrend.Refresh
	Else
		MsgBox "objTrend is nothing",vbExclamation,GetProjectTitle
	End If
End Sub

Public Sub SymbolLoading()
	Set objTrend = GetSynopticObject.GetSubObject("Trend").GetObjectInterface
End Sub
```

ShowPagePrevBtn, TrendCmdTarget Property

Syntax ShowPagePrevBtn = _Boolean

Description This property permits the Trend object's Page Previous command button to be displayed or hidden.

Parameter	Description
None	None

Result Boolean

Example:

```
Dim objTrend As TrendCmdTarget
Public Sub Click()
	If Not objTrend Is Nothing Then
		MsgBox "objTrend 's ShowPagePrevBtn is " & objTrend.ShowPagePrevBtn,vbInformation,GetProjectTitle
		objTrend.ShowPagePrevBtn = Not objTrend.ShowPagePrevBtn
		objTrend.Refresh
	Else
		MsgBox "objTrend is nothing",vbExclamation,GetProjectTitle
	End If
End Sub

Public Sub SymbolLoading()
	Set objTrend = GetSynopticObject.GetSubObject("Trend").GetObjectInterface
End Sub
```

ShowPauseRunBtn, TrendCmdTarget Property

Syntax ShowPauseRunBtn = _Boolean

Description This property permits the Trend object's Run/Pause command button to be displayed or hidden.

Parameter	Description
None	None

Result Boolean

Example:

```
Dim objTrend As TrendCmdTarget
Public Sub Click()
        If Not objTrend Is Nothing Then
                MsgBox "objTrend 's ShowPauseRunBtn is " & objTrend.ShowPauseRunBtn,vbInformation,GetProjectTitle
                objTrend.ShowPauseRunBtn = Not objTrend.ShowPauseRunBtn
                objTrend.Refresh
        Else
                MsgBox "objTrend is nothing",vbExclamation,GetProjectTitle
        End If
End Sub

Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetSubObject("Trend").GetObjectInterface
End Sub
```

ShowPen, TrendCmdTarget Property

Syntax ShowPen = _Boolean

Description This property sets or returns the option for enabling or disabling the showing of the Trend's pens. Accepts a boolean value.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetSubObject("TrendObj").GetObjectInterface
End Sub
Public Sub Click()
        objTrend.ShowPen = Not objTrend.ShowPen
```

```
        objTrend.Refresh
End Sub
```

ShowPenLabels, TrendCmdTarget Property

Syntax ShowPenLabels(_lpszPenName) = Boolean

Description This property consents you to read and set the display status of the value labels in correspondence to each graphical point. In cases where this property has been set with a new value, you will need to used the "Refresh" method to update the object graphically.

Parameter	Description
lpszPenName as string	Pen name

Result Boolean

Example:

```
Dim objTrend As TrendCmdTarget
Public Sub SymbolLoading()
        Dim i As Long
        Dim sPenList() As String
        Dim nPenSelected As Long

        If obj Is Nothing Then Set obj =
        GetSynopticObject.GetSubObject("objTrend").GetObjectInterface
        Erase sPenList
        For i = 0 To obj.GetPensNumber - 1
                ReDim Preserve sPenList(i)
                sPenList(i) = obj.GetPenNameFromList(i)
        Next
        nPenSelected = ShowPopupMenu(sPenList)
        If nPenSelected >= 0 Then
                obj.ShowPenLabels(sPenList(nPenSelected)) = Not
                obj.ShowPenLabels(sPenList(nPenSelected))
                obj.Refresh
        End If
End Sub
```

ShowPenPoints, TrendCmdTarget Property

Syntax ShowPenPoints(_lpszPenName) = Boolean

Description When enabled, this property allows all the points to be displayed for the selected line.

Parameter	Description
lpszPenName As String	Pen's name

Result Integer

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.ShowPenPoints(objTrend.GetPenNameFromList(0))  =  Not
        objTrend.ShowPenPoints(objTrend.GetPenNameFromList(0))
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

ShowPrevBtn, TrendCmdTarget Property

Syntax ShowPrevBtn = _Boolean

Description This property permits the Trend object's Previous command button to be displayed or hidden.

Parameter	Description
None	None

Result Boolean

Example:

```
Dim objTrend As TrendCmdTarget
Public Sub Click()
        If Not objTrend Is Nothing Then
                MsgBox "objTrend 's ShowPrevBtn is " &
                objTrend.ShowPrevBtn,vbInformation,GetProjectTitle
                objTrend.ShowPrevBtn = Not objTrend.ShowPrevBtn
                objTrend.Refresh
        Else
                MsgBox "objTrend is nothing",vbExclamation,GetProjectTitle
        End If
End Sub

Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetSubObject("Trend").GetObjectInterface
End Sub
```

ShowPrevPointBtn, TrendCmdTarget Property

Syntax ShowPrevPointBtn = _Boolean

Description This property makes the scroll cursor button visible/not visible which is used for moving the cursor to the previous point of the curve displayed in the Data Analysis.
This property is only managed by the Data Analysis and not the Trend object.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objDataAnalysis As TrendCmdTarget
Public Sub Click()

        If Not objDataAnalysis Is Nothing Then
                objDataAnalysis.ShowPrevPointBtn= Not
                objDataAnalysis.ShowPrevPointBtn
        End If
End Sub

Public Sub SymbolLoading()
        Set objDataAnalysis =
        GetSynopticObject.GetSubObject("objDataAnalysis").GetObjectInterface
End Sub
```

ShowPrintBtn, TrendCmdTarget Property

Syntax ShowPrintBtn = _Boolean

Description This property permits the Trend object's Print command button to be displayed or hidden.

This property is not supported in Windows CE.(If set always returns 'false')

Parameter	Description
None	None

Result Boolean

Example:

```
Dim objTrend As TrendCmdTarget
Public Sub Click()
        If Not objTrend Is Nothing Then
                MsgBox "objTrend 's ShowPrintBtn is " &
                objTrend.ShowPrintBtn,vbInformation,GetProjectTitle
                objTrend.ShowPrintBtn = Not objTrend.ShowPrintBtn
                objTrend.Refresh
        Else
                MsgBox "objTrend is nothing",vbExclamation,GetProjectTitle
        End If
End Sub

Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetSubObject("Trend").GetObjectInterface
End Sub
```

ShowSavedValues, TrendCmdTarget Property

Syntax ShowSavedValues(_lpszPenName) = Boolean

Description When enabled this property allows the values saved with the SaveAllCurrentValue or SaveCurrentValue functions to be displayed for the pen specified with the lspszPenName parameter. The pen's saved values are displayed by means of a hatched line.

Parameter	Description
lpszPenName As String	Pen's name

Result Integer

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.ShowSavedValues(objTrend.GetPenNameFromList(0))        =        Not
        objTrend.ShowSavedValues(objTrend.GetPenNameFromList(0))
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

ShowTime, TrendCmdTarget Property

Syntax ShowTime = _Boolean

Description This property sets or returns the option for enabling or disabling the showing of the Trend's time. Accepts a boolean value.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetSubObject("TrendObj").GetObjectInterface
End Sub
Public Sub Click()
        objTrend.ShowTime = Not objTrend.ShowTime
        objTrend.Refresh
End Sub
```

ShowTimeFrameBtn, TrendCmdTarget Property

Syntax ShowTimeFrameBtn = _Boolean

Description This property permits you to display or hide the buttons used for selecting time frames of data represented by the Data Analysis object. In cases where this property has been set with a new value, you will need to used the "Refresh" method to update the object graphically.

Parameter	Description
None	None

Result Boolean

Example:

```
Dim obj As TrendCmdTarget
Public Sub Click()
        If obj Is Nothing Then Set obj =
        GetSynopticObject.GetSubObject("objDataAnalysis").GetObjectInterface
        obj.ShowTimeFrameBtn = Not obj.ShowTimeFrameBtn
        obj.Refresh
End Sub
```

ShowTitle, TrendCmdTarget Property

Syntax ShowTitle = _Boolean

Description This property sets or returns the option for enabling or disabling the showing of the Trend's title. Accepts a boolean value.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetSubObject("TrendObj").GetObjectInterface
End Sub
Public Sub Click()
        objTrend.ShowTitle = Not objTrend.ShowTitle
        objTrend.Refresh
End Sub
```

ShowXGrid, TrendCmdTarget Property

Syntax ShowXGrid = _Boolean

Description This property sets or returns the option for enabling or disabling the showing of the X grid. Accepts a boolean value.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetSubObject("TrendObj").GetObjectInterface
End Sub
Public Sub Click()
        objTrend.ShowXGrid = Not objTrend.ShowXGrid
        objTrend.Refresh
End Sub
```

ShowYGrid, TrendCmdTarget Property

Syntax ShowYGrid = _Boolean

Description This property sets or returns the option for enabling or disabling the showing of the Y grid. Accepts a boolean value.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetSubObject("TrendObj").GetObjectInterface
End Sub
Public Sub Click()
        objTrend.ShowYGrid = Not objTrend.ShowYGrid
        objTrend.Refresh
End Sub
```

ShowZoomBtn, TrendCmdTarget Property

Syntax ShowZoomBtn = _Boolean

Description This property displays or hides the Zoom button. In cases where this property has been set with a new value, you will need to used the "Refresh" method to update the object graphically.

Parameter	Description
None	None

Result Boolean

Example:

```
Dim obj As TrendCmdTarget
Public Sub Click()
        If obj Is Nothing Then Set obj =
        GetSynopticObject.GetSubObject("objDataAnalysis").GetObjectInterface
        obj.ShowZoomBtn = Not obj.ShowZoomBtn
        obj.Refresh
End Sub
```

StartNewFile , TrendCmdTarget Property

Syntax StartNewFile = _Boolean

Description This property enables the creation of a new file, in CSV format, each time the project goes into run mode. In order for this to work properly, you need to save the new settings in the trend's configuration file. When set with the True boolean value, the system will load the new configurations and initialize the new file at the first project startup.

Parameter	Description
None	None

Result Long

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.StartNewFile = True
        objTrend.MaxFileLength = CInt(InputBox("Max file length"))
        objTrend.SaveExtSettings
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

StatAverageValue, TrendCmdTarget Property

Syntax StatAverageValue(lpszPenName) = Double

Description This property sets or returns the average value of a pen's samples calculated on the values displayed in the trend value. Any value changes will remain valid until the next trend refresh takes place where the system calculates the average value again.

Parameter	Description
None	None

Result Double

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dbug.Print objTrend.StatAverageValue(objTrend.GetPenNameFromList(0))
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

StatMaxValue, TrendCmdTarget Property

Syntax StatMaxValue(lpszPenName) = Double

Description This property sets or returns the maximum value of a pen's samples calculated on the values displayed in the trend window. Any value changes will remain valid until the next trend refresh takes place where the system calculates the maximum value again.

Parameter	Description
None	None

Result Double

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dbug.Print objTrend.StatMaxValue(objTrend.GetPenNameFromList(0))
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

StatMinValue, TrendCmdTarget Property

Syntax StatMinValue(lpszPenName) = Double

Description This property sets or returns a pen's minimum sample value calculated on the values displayed in the trend window. Any value changes will remain valid until the next trend refresh takes place where the system recalculated the minimum value again.

Parameter	Description
None	None

Result Double

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dbug.Print objTrend.StatMinValue(objTrend.GetPenNameFromList(0))
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

TimeBrushColor, TrendCmdTarget Property

Syntax TimeBrushColor = _Long

Description This property sets or returns the back color code of the Trend's time area.

Parameter	Description
None	None

Result Long

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.TimeBrushColor = RGB(255,0,0)
        objTrend.Refresh
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

TimeBrushVisible, TrendCmdTarget Property

Syntax TimeBrushVisible = _Boolean

Description This property sets or returns the visible property of the trend's time area back color.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.TimeBrushVisible = Not objTrend.TimeBrushVisible
        objTrend.Refresh
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

TimeFrameBtnColor, TrendCmdTarget Property

Syntax TimeFrameBtnColor = _Long

Description This property allows you to read and set the colour used for displaying buttons for the data timeframe representations for the Data Analysis object. In case where the property have been set with a new value, you will need to use the "Refresh" method for updating the object graphically.

Parameter	Description
None	None

Result Long

Example:

```
Dim obj As TrendCmdTarget
Public Sub Click()
        Dim lColor As Long

        If obj Is Nothing Then Set obj =
        GetSynopticObject.GetSubObject("objDataAnalysis").GetObjectInterface
        If ChooseColor(lColor) Then
                obj.TimeFrameBtnColor = lColor
                obj.Refresh
        End If
End Sub
```

TimeScale, TrendCmdTarget Property

Syntax TimeScale= _Byte

Description This property sets or returns the value corresponding to the "Time Scale" option so that the time axis adjusts according to the following list:

0 = Adjust to Values
1 = Absolute Range
2 = Adjust to Range

This property is only managed by the Data Analysis object and not the Trend object.

Parameter	Description
None	None

Result Byte

Example:

```
Option Explicit

Enum eTimeScale
        eAdjustToValues = 0
        eAbsoluteRange = 1
        eAdjustToRange = 2
End Enum
Dim objDataAnalysis As TrendCmdTarget
Public Sub Click()

        If Not objDataAnalysis Is Nothing Then
objDataAnalysis.TimeScale = eAdjustToValues
        objDataAnalysis.Requery
        End If
End Sub

Public Sub SymbolLoading()
        Set objDataAnalysis =
        GetSynopticObject.GetSubObject("objDataAnalysis").GetObjectInterface
End Sub
```

TimeTextColor, TrendCmdTarget Property

Syntax TimeTextColor = _Long

Description This property sets or returns the trend's time area text colour display property.

Parameter	Description
None	None

Result Long

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.TimeTextColor = RGB(255,0,0)
        objTrend.Refresh
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

TrendBrushColor, TrendCmdTarget Property

Syntax TrendBrushColor = _Long

Description This property sets or returns the back color code of the trend area.

Parameter	Description
None	None

Result Long

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.TrendBrushColor = RGB(255,0,0)
        objTrend.Refresh
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

TrendBrushVisible, TrendCmdTarget Property

Syntax TrendBrushVisible = _Boolean

Description This property sets or returns the Visible property of the Trend area's back color.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.TrendBrushVisible = Not objTrend.TrendBrushVisible
        objTrend.Refresh
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

TrendRunningType, TrendCmdTarget Property

Syntax TrendRunningType = _Integer

Description This property returns, in read only, the trend's execution type whether Run/Stop or Run only and Stop only.

Returns a integer value with the following meanings:
0=Run/Stop
1=Run Only
2=Stop Only

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Debug.Print objTrend.TrendRunningType
        objTrend.Refresh
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

VariableAddValue, TrendCmdTarget Property

Syntax VariableAddValue = _String

Description This property sets or returns the recording variable under the Trend's command. When the "Add. Val" is used the trend will not record on a time basis, but will execute a recording every time that this variable is set to the 1 value. When the recording has been executed Movicon will return the variable to the 0 value. When resetting the variable with a nothing string, the trend will return to record on a time basis.
Before using this property for changing the associated variable you will need to momentarily suspend the recording by setting the Recording value to False.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim sName As String
        Debug.Print "VariableAddValue = " & objTrend.VariableAddValue
        GetVariableNameFromList(sName)
        objTrend.Recording = False
        objTrend.VariableAddValue  = sName
        objTrend.Recording = True
        Debug.Print "VariableAddValue = " & objTrend.VariableAddValue
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

VariableCursorPosIn, TrendCmdTarget Property

Syntax VariableCursorPosIn = _String

Description This property sets or returns the name of the variable associated to the Trend's "Cursor In" function.
To reset the variable you will need to momentarily suspend the recording by setting the Recording property to False.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim sName As String
        Debug.Print "VariableCursorPosIn = " & objTrend.VariableCursorPosIn
        GetVariableNameFromList(sName)
        objTrend.Recording = False
        objTrend.VariableCursorPosIn = sName
        objTrend.Recording = True
        Debug.Print "VariableCursorPosIn = " & objTrend.VariableCursorPosIn
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

VariableCursorPosOut, TrendCmdTarget Property

Syntax VariableCursorPosOut = _String

Description This property sets or returns the name of the variable associated to the Trend's "Cursor Out" function. To set this variable again you will need to momentarily suspend the recording by setting the Recording property to False.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim sName As String
        Debug.Print "VariableCursorPosOut  = " & objTrend.VariableCursorPosOut
        GetVariableNameFromList(sName)
        objTrend.Recording = False
        objTrend.VariableCursorPosOut = sName
        objTrend.Recording = True
        Debug.Print "VariableCursorPosOut = " & objTrend.VariableCursorPosOut
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

VariableEnabling, TrendCmdTarget Property

Syntax VariableEnabling = _String

Description This property sets or returns the name of the variable associated to the trend's "Enable" function. To set a new variable you will need to momentarily suspend the recording by setting the Recording to False.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim sName As String
```

```
        Debug.Print "VariableEnabling = " & objTrend.VariableEnabling
        GetVariableNameFromList(sName)
        objTrend.Recording = False
        objTrend.VariableEnabling = sName
        objTrend.Recording = True
        Debug.Print "VariableEnabling = " & objTrend.VariableEnabling
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

VariableFreezedMode, TrendCmdTarget Property

Syntax VariableFreezedMode = _String

Description This property sets or returns the name of the variable associated to the trend's "Start/Stop " function. To set a new variable you will need to momentarily suspend the recording by setting the Recording property to False.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim sName As String
        Debug.Print "VariableFreezedMode = " & objTrend.VariableFreezedMode
        GetVariableNameFromList(sName)
        objTrend.Recording = False
        objTrend.VariableFreezedMode = sName
        objTrend.Recording = True
        Debug.Print "VariableFreezedMode = " & objTrend.VariableFreezedMode
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

VariableResetAllValues, TrendCmdTarget Property

Syntax VariableResetAllValues = _String

Description This property sets or returns the name of the variable associated to the trend's "Reset" function. To reset a new variable you will need to momentarily suspend the recording by setting thye Recording property to False.

Parameter	Description

None	None

Result String

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
	Dim sName As String
	Debug.Print "VariableResetAllValues = " & objTrend.VariableResetAllValues
	GetVariableNameFromList(sName)
	objTrend.Recording = False
	objTrend.VariableResetAllValues = sName
	objTrend.Recording = True
	Debug.Print "VariableResetAllValues = " & objTrend.VariableResetAllValues
End Sub
Public Sub SymbolLoading()
	Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

VariableScrollEnd, TrendCmdTarget Property

Syntax VariableScrollEnd = _String

Description This property sets or returns the name of the variable associated to the trend's "End" function. To set a new variable you will need to momentarily suspend the Recording property to False.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
	Dim sName As String
	Debug.Print "VariableScrollEnd = " & objTrend.VariableScrollEnd
	GetVariableNameFromList(sName)
	objTrend.Recording = False
	objTrend.VariableScrollEnd = sName
	objTrend.Recording = True
	Debug.Print "VariableScrollEnd = " & objTrend.VariableScrollEnd
End Sub
Public Sub SymbolLoading()
	Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

VariableScrollNext, TrendCmdTarget Property

Syntax VariableScrollNext = _String

Description This property sets or returns the name of the variable associated to the trend's "Next" function. To set a new variable you will need to suspend the recording by setting the Recording property to False.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim sName As String
        Debug.Print "VariableScrollNext = " & objTrend.VariableScrollNext
        GetVariableNameFromList(sName)
        objTrend.Recording = False
        objTrend.VariableScrollNext = sName
        objTrend.Recording = True
        Debug.Print "VariableScrollNext = " & objTrend.VariableScrollNext
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

VariableScrollNextPage, TrendCmdTarget Property

Syntax VariableScrollNextPage = _String

Description This property sets or returns the name of the variable associated to the trend's "Next Page" function. To set a new variable you will need to momentarily suspend the recording by setting the Recording property to False.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim sName As String
        Debug.Print "VariableScrollNextPage = " & objTrend.VariableScrollNextPage
        GetVariableNameFromList(sName)
        objTrend.Recording = False
```

```
        objTrend.VariableScrollNextPage = sName
        objTrend.Recording = True
        Debug.Print "VariableScrollNextPage = " & objTrend.VariableScrollNextPage
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

VariableScrollPrev, TrendCmdTarget Property

Syntax VariableScrollPrev = _String

Description This property sets or returns the name of the variable associated to the trend's "Previous" function. To set a new variable you will need to momentarily suspend the Recording property to False.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim sName As String
        Debug.Print "VariableScrollPrev = " & objTrend.VariableScrollPrev
        GetVariableNameFromList(sName)
        objTrend.Recording = False
        objTrend.VariableScrollPrev = sName
        objTrend.Recording = True
        Debug.Print "VariableScrollPrev = " & objTrend.VariableScrollPrev
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

VariableScrollPrevPage, TrendCmdTarget Property

Syntax VariableScrollPrevPage = _String

Description This property sets or returns the name of the variable associated to the trend's "Previous Page" function. To set a new variable you will need to momentarily suspend the recording by setting the Recording property to False.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim sName As String
        Debug.Print "VariableScrollPrevPage = " & objTrend.VariableScrollPrevPage
        GetVariableNameFromList(sName)
        objTrend.Recording = False
        objTrend.VariableScrollPrevPage = sName
        objTrend.Recording = True
        Debug.Print "VariableScrollPrevPage = " & objTrend.VariableScrollPrevPage
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

VariableScrollStart, TrendCmdTarget Property

Syntax VariableScrollStart = _String

Description This property sets or returns the name of the variable associated to the trend's "Start" function. To set a new variable you will need to momentarily suspend the recording by setting the Recording property to False.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Dim sName As String
        Debug.Print "VariableScrollStart = " & objTrend.VariableScrollStart
        GetVariableNameFromList(sName)
        objTrend.Recording = False
        objTrend.VariableScrollStart = sName
        objTrend.Recording = True
        Debug.Print "VariableScrollStart = " & objTrend.VariableScrollStart
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

Vertical, TrendCmdTarget Property

Syntax Vertical = _Boolean

Description This property sets or returns the configuration of the trend object's vertical or horizontal style. Accepts a boolean value.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetSubObject("TrendObj").GetObjectInterface
End Sub
Public Sub Click()
        objTrend.Vertical = Not objTrend.Vertical
        objTrend.Refresh
End Sub
```

ViewSamples, TrendCmdTarget Property

Syntax ViewSamples = _Long

Description This read only property returns the number of samples displayed in the Trend window.

Parameter	Description
None	None

Result Long

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        Debug.Print "ViewSamples  = " & objTrend.ViewSamples
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

Visible, TrendCmdTarget Property

Syntax Visible(_lpszPenName) = _Boolean

Description This property enables or disables the visibility of the pen indicated as parameter.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
	objTrend.Visible(objTrend.GetPenNameFromList(0)) = Not
	objTrend.Visible(objTrend.GetPenNameFromList(0))
End Sub
Public Sub SymbolLoading()
	Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

WeekBtnText, TrendCmdTarget Property

Syntax WeekBtnText = _String

Description Questa proprietà imposta o restituisce il testo che dovrà apparire sul "Pulsante Settimana" della barra dei pulsanti per la selezione dell''intervallo di tempo dell'oggetto Data Analysis. When nothing has been specified, Movicon will use the default text instead.

Parameter	Description
None	None

Result String

Example:

```
Dim objDataAnalysis As TrendCmdTarget
Public Sub Click()
	If Not objDataAnalysis Is Nothing Then
		MsgBox "objDataAnalysis's WeekBtnText is " &
		objDataAnalysis.WeekBtnText,vbInformation,GetProjectTitle
		objDataAnalysis.WeekBtnText = "Week"
		objDataAnalysis.Refresh
	Else
		MsgBox "objDataAnalysis is nothing",vbExclamation,GetProjectTitle
	End If
End Sub

Public Sub SymbolLoading()
	Set objDataAnalysis =
	GetSynopticObject.GetSubObject("DataAnalysis").GetObjectInterface
End Sub
```

XGridColor, TrendCmdTarget Property

Syntax XGridColor = _Long

Description This property sets the code of the color used for the X Grid.

Parameter	Description
None	None

Result Long

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.XGridColor = RGB(245,0,0)
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

XGridLogarithmic, TrendCmdTarget Property

Syntax XGridLogarithmic = _Boolean

Description This property enables the logarithmic scale for the Trend area's X grid.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.XGridColor = Not objTrend.XGridColor
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

XGridUseNormalLine, TrendCmdTarget Property

Syntax XGridUseNormalLine = _Boolean

Description This property sets or returns the line type, normal or dashed, for the trend area's X grid.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.XGridUseNormalLine = Not objTrend.XGridUseNormalLine
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

XY, TrendCmdTarget Property

Syntax XY = _Boolean

Description This property sets or returns the trend's graphic style, whether XY type or Horizontal type. The trend's graphic vertical style cannot be set with this property and therefore you will need to use the "Vertical" property to do so.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.XY = Not objTrend.XY
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

YearBtnText, TrendCmdTarget Property

Syntax YearBtnText = _String

Description This property sets or returns the text to display on the "Year Button" on the button bar for selecting time ranges in the Data Analysis object. When nothing has been specified, Movicon will use the default text instead.

Parameter	Description
None	None

Result String

Example:

```
Dim objDataAnalysis As TrendCmdTarget
Public Sub Click()
	If Not objDataAnalysis Is Nothing Then
		MsgBox "objDataAnalysis's YearBtnText is " & objDataAnalysis.YearBtnText,vbInformation,GetProjectTitle
		objDataAnalysis.YearBtnText = "Year"
		objDataAnalysis.Refresh
	Else
		MsgBox "objDataAnalysis is nothing",vbExclamation,GetProjectTitle
	End If
End Sub

Public Sub SymbolLoading()
	Set objDataAnalysis = GetSynopticObject.GetSubObject("DataAnalysis").GetObjectInterface
End Sub
```

YGridColor, TrendCmdTarget Property

Syntax YGridColor = _Long

Description This property sets the code of the color used for the Y grid.

Parameter	Description
None	None

Result Long

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
	objTrend.YGridColor = RGB(245,0,0)
End Sub
Public Sub SymbolLoading()
	Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
```

```
End Sub
```

YGridLogarithmic, TrendCmdTarget Property

Syntax YGridLogarithmic = _Boolean

Description This property enables the logarithmic scale for the trend area's Y grid.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.YGridColor = Not objTrend.YGridColor
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

YGridUseNormalLine, TrendCmdTarget Property

Syntax YGridUseNormalLine = _Boolean

Description This property sets or returns the line type, normal or dashed, for the Trend area's Y grid.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.YGridUseNormalLine = Not objTrend.YGridUseNormalLine
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

ZoomBtnText, TrendCmdTarget Property

Syntax ZoomBtnText = _String

Description This property sets or returns a text for the Trend/Data Analysis object's zoom command button. In cases where this property is set with a new value, you will need to use the "Refresh" method to update the object graphically.

Parameter	Description
None	None

Result String

Example:

```
Dim obj As TrendCmdTarget
Public Sub Click()
        If obj Is Nothing Then Set obj =
        GetSynopticObject.GetSubObject("objTrend").GetObjectInterface
        If obj.MeasureBtnText <> "" Then
                obj.ZoomBtnText = ""
        Else
                obj.ZoomBtnText = "ZOOM"
        End If
        obj.Refresh
End Sub
```

ZoomMode, TrendCmdTarget Property

Syntax ZoomMode = _Boolean

Description This property sets or returns the Trend area's zoom mode. When the zoom mode is activated you can enlarge a portion of the trend area by selecting the area with the mouse. When the zoom mode is deactivated the trend area will remain enlarged and it can be scrolled with the cursor. Use the ResetZoom() function to return its initial conditions.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Dim objTrend As TrendCmdTarget
Public Sub Click()
        objTrend.ZoomMode = Not objTrend.ZoomMode
End Sub
Public Sub SymbolLoading()
        Set objTrend = GetSynopticObject.GetAbsoluteSubObject("Trend").GetObjectInterface
End Sub
```

55. UIInterface

AlphaNumericEntry, UIInterface Function

Syntax AlphaNumericEntry(_lpszVariableName, _nMaxChars)

Description Shows a alphanumeric display for setting alphanumeric values with the mouse or the touchscreen to ve associated to the Movicon variable. Meant for PCs without keyboards.

Parameter	Description
lpszVariableName As String	Name of the Movicon variable.
nMaxChars As Integer	Optional. Value indicating the maximum number of digital characters.

Result Boolean

Example:

```
Sub Main
        ...
        AlphaNumericEntry("Setpoint_23", 3)
        ...
End Sub
```

ChooseColor, UIInterface Function

Syntax ChooseColor(_pColor)

Description This function displays a window showing a range of colors. The selecting of a color closes the window and assigns the selected color's RGB code to the pColor parameter.
When existing from the color palette with the Esc key or by clicking it with the mouse, a False value will be returned and the pColor parameter will return to zero.
The returned True value indicates that a color has been selected while the False value indicates the color selection was not successful.

This function is not supported in Windows CE.(If set always returns 'false')

Parameter	Description
pColor As Long	Color selected code.

Result Boolean

Example:

```
Public Sub Click()
```

```
        Dim pColor As Long
        'View select window
        ChooseColor(pColor)
        MsgBox("Color=" & pColor, ,"ChooseColor")
        'GlobalContainerName enabled
        ContainerDoc.BackColor = pColor
End Sub
```

DoSomeEvents, UIInterface Function

Syntax DoSomeEvents(_nMaxEvents)

Description This function, different to the DoEvent function, permits you to leave part of the basic resources for executing other functions within the same code. Normally, once a program loop enters into execution where the DoEvents are being used it becomes no longer possible to execute other functions until the loop has completed. However, when using the DoSomeEvents function you can specify, through the nMaxEvents parameter, the number of loops to be executed before letting other operations to be executed. This function is useful in basic code events associated to symbols which are executed in a long time. Using this function Movicon leaves a part of the events execution time to the User Interface.

Parameter	Description
nMaxEvents As Integer	Number of loops to be executed.

Result Boolean

Example:

```
Sub Main
        While Not CBool(GetVariableValue("Bit"))
        DoSomeEvents(100)
        Wend
        Debug.Print "End loop"
End Sub
```

EditRuntimeUsers, UIInterface Function

Syntax EditRuntimeUsers()

Description Permits you to edit the runtime Users by using the appropriate Movicon tool.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Main
        ...
```

```
        EditRuntimeUsers
        ...
End Sub
```

ExecuteCommand, UIInterface Function

Syntax ExecuteCommand(_lpszCommand)

Description This function allows you to execute commands from the Movicon Comand List in Scripts. The lpszCommand parameter must return a string containing the command type and the parameters for executing it. The syntax must be:

"<CommandType ...command parameters ...''>CommandType</CommandType>"

When the single quote (') character is inserted directly with command's parameter, Movicon may interpret this character as the closing of the paramater's value. For instance, the following line is not correct:

```
Public Sub Click()
  'Show tooltip
  ExecuteCommand("<CommandType action='1' topic='Hello
'World''>Help</CommandType>")
End Sub
```

A solution to remedy this would be to use "'" instead of "'":

```
Public Sub Click()
  'Show tooltip
  ExecuteCommand("<CommandType action='1' topic='Hello
&apos;World&apos;'>Help</CommandType>")
End Sub
```

The command parameters may be:

Variable Commands
variable='VariableName' a SecondVariable='DestinationVariable' action='ActionType' strobe='StrobeValue' value='VariableValue' max='MaxVal' min='MinVal' chars='MaxChars' ' alis='' pwd=''

where:
ActionType
0 Set Variable
1 Reset Variable
2 Toggle Variable
3 Strobe Variable
4 Increase Variable
5 Decrease Variable
6 Alphanumeric Pad
7 Numeric Pad
8 Append Value
9 Remove Value
10 Swap Plus-Minus
11 Append Decimal Mode ON-OFF
12 Move Value
13 Reset Statistics
14 Move Min. Value
15 Move Max. Value
16 Move Average Value
17 = Set String Table Value
18 = Set Screen Alias

Screen Commands
synoptic='SynopticName' action='ActionType' monitor='0'

parameter='ParameterList' x='Xposition' y='Yposition' width='Width' height='Height' Caption='1' Border='1' Resizeable='0' SysMenu='0' MinimizeBox='0' MaximizeBox='0' KeepPrintProportions='0' PageW='-1' PageH='-1' LMargin='-1' RMargin='-1' TMargin='-1' BMargin='-1'

where:
ActionType
0 Open Normal
1 Open Modal
2 Open Frame
3 Open Safe
4 Print
5 Close
6 Execute Synapse
7 Open Next
8 Open Prev
9 Capture and Print
10 Capture and Save

Parameterlist
parametervalue1, parametervalue2,, ..., ParametervalueN

Script Commands
script='ScriptName' action='ActionType' parameters='ParameterList' newInstAllowed='0' timeout="TimeoutValue"

whrere:
ActionType
0 Run Normal
1 Run Syncro
2 Run Safe
3 Stop
4 Unload

Parameterlist
parametervalue1, parametervalue2,, ..., ParametervalueN

User Commands
action='ActionType' level='UserLevel'

where:
ActionType
0 LogOn
1 LogOff
2 Edit

Report Commands
dlr='Data LoggerName' action='ActionType' x='XPosition' y='YPosition' width='Width' height='Height' Toolbar='0' GroupTree='true/false'' TemplateFile='' DestinationFile='' Query='' ReferencePeriod='0'' ExportFormat='0'' SelectDate='0'' Sep='59' MaxPages='0' LeftMargin='-1' RightMargin='-1' TopMargin='-1' BottomMargin='-1' PrintDlg='0' Landscape='0' Printer='' Recipient='' EmbRep=''

where:
ActionType
0 = View Synchronouse
1 = Print Synchronouse
2 = View
3 = Print
4 Move First
5 Move Last
6 Move Prev
7 Move Next
8 Activate
9 Save
10 Delete
11 Requery
12 Execute Query

13 Export
14 Data Analysis
15 View Textual Report
16 Print Textual Report
17 Save Textual Report
18 Append Textual Report
19 = Export Recipe
20 = Import Recipe
21 = Export and Send Email
22 = View Embedded Report
23 = Print Embedded Report
24 = Save Embedded Report
25 = Send Embedded Report
26 = Read

ReferencePeriod
0 = None
1 = Today
2 = Yesterday and Today
3 = Current Week
4 = Current Month
5 = Current Year
6 = Last 7 Days
7 = Last 30 Days
8 = Last 60 Days
9 = Last 90 Days
10 = Last Year
11 = Last 2 Years
12 = Last 5 Years
13 = Last 10 Years

ExportFormat
0 = Nessuno
1 = Pdf
2 = Html
3 = Txt
4 = Csv
5 = Xls
6 = Mht
7 = Rtf
8 = Jpeg

SelectDate
0 = the date selection window will not show
1 = window for selecting dates to apply date filters will show

Menu Commands
menu='MenuName' x='-1' y='-1'

System Commands
action='ActionType' parameters='Parameter' workingpath='' timeout='TimeoutValue'

Where:
ActionType
0 Shut Down OS
1 Shut Down App
2 Launch App
3 Launch App and Wait
4 Play Sound File
5 Beep
6 Speack
7 Reboot OS
8 = Show or Hide the Output Window
9 = Wait Time

Parameter
Application Name

Help Commands
action='ActionType' topic='Topic'

where:
ActionType
0 = Topic
1 = Tooltip popup

Change Language
Language='Language'

Alarm Commands
action='ActionType' AreaFilter='' Report='Report' Toolbar='true/false' GroupTree='true/false' Period='Period' Duration='Duration' Date='Date' TemplateFile='' DestinationFile='' Query='' MaxPages='10' PageHeight='-1' PageWidth='-1' LeftMargin='-1' RightMargin='-1' TopMargin='-1' BottomMargin='-1' PrintDlg='0' Landscape='0' Printer='' Recipient=''

where:
ActionType
0 Ack All
1 Reset All
2 Toggle Sound
3 View Report
4 Print Report
5 Export Report
6 View Textual Report
7 Print Textual Report
8 Save Textual Report
9 Append Textual Report
10 = Export and Send Email
11 View Embedded Report
12 Print Embedded Report
13 Save Embedded Report
14 Send Embedded Report
15 Reset Statistics

Report
OrderByDate
OrderByDuration
GroupByFrequency
GroupByThreshold

ReferencePeriod
0 = Period
1 = Today
2 = Yesterday or Today
3 = Current week
4 = Current month
5 = Current year
6 = Last 7 days
7 = Last 30 days
8 = Last 60 days
9 = Last 90 days
10 = Last 1 years
11 = Last 2 years
12 = Last 5 years
13 = Last 10 years

Event Command
eventname='Event1'

Parameter	Description

lpszCommand As String	Command Line.

Result Boolean

Example1:

```
Public Sub Click()
        'Set VAR00001 = 1
        ExecuteCommand("<CommandType variable='VAR00001' action='0'
        strobe='0' value='1' max='100' min='0' _ chars=''>Variable</CommandType>")
End Sub
```

Example2:

```
Public Sub Click()
        'Open Synopric MDI
        ExecuteCommand("<CommandType synoptic='LayOut' action='0' monitor='0'
        parameter='' x='-1' y='-1' width='0' height='0' Caption='true' Border='true'
        Resizeable='false' SysMenu='false' MinimizeBox='false'
        MaximizeBox='false'KeepPrintProportions='0' PageW='-1' PageH='-1' LMargin='-
        1' RMargin='-1' TMargin='-1' BMargin='-1'>Synoptic</CommandType>")
End Sub
```

Example3:

```
Public Sub Click()
        'Execute Script Normal
        ExecuteCommand("<CommandType script='Script1' action='0'
        parameters='1,2,3' newInstAllowed='0' timeout='3000'
        >Script</CommandType>")
End Sub
```

Example4:

```
Public Sub Click()
        'LogOn User
        ExecuteCommand("<CommandType action='0'
        level='5'>Users</CommandType>")
End Sub
```

Example5:

```
Public Sub Click()
        'Data Logger View Report
        ExecuteCommand("<CommandType dlr='Data Logger' action='2' x='-1' y='-1'
        width='0' height='0' Toolbar='true' GroupTree='true'
        >Report</CommandType>")
End Sub
```

Example6:

```
Public Sub Click()
        'Data Logger View Textual Report
        ExecuteCommand("<CommandType dlr='Data Logger' action='15'
        TemplateFile='Template.txt' DestinationFile='' Query='' MaxPages='0'
        >Report</CommandType>")
End Sub
```

Example7:

```
Public Sub Click()
        'Launch calc.exe
        ExecuteCommand("<CommandType action='2' parameters='calc.exe'
        timeout='5000'>System</CommandType>")
        'Close Supervisor
        ExecuteCommand("<CommandType action='1' parameters=''
        timeout='5000'>System</CommandType>")
End Sub
```

Example8:

```
Public Sub Click()
```

```
        'Show tooltip
        ExecuteCommand("<CommandType action='1' topic='Welcome to
        Movicon'>Help</CommandType>")
End Sub
```

Example9:

```
Public Sub Click()
        'Change language to English, that has been previously defined in the String
        Table
        ExecuteCommand("<CommandType
        action='English'>Language</CommandType>")
End Sub
```

Example10:

```
Public Sub Click()
        'AckAll Alarms
        ExecuteCommand("<CommandType action='0' Report='' Toolbar='true'
        GroupTree='false' Period='' Duration='' Date=''>Alarm</CommandType>")
End Sub
```

Example11:

```
Public Sub Click()
        'Alarms View Textual Report
        ExecuteCommand("<CommandType action='6' TemplateFile='Template.txt'
        DestinationFile='' Query='' MaxPages='0' >Alarm</CommandType>")
End Sub
```

GetLastActiveSynoptic, UIInterface Function

Syntax GetLastActiveSynoptic

Description Returns a string containing the name of the last screen opened for viewing.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        ...
        Dim sResult As String
        sResult = GetLastActiveSynoptic
        Debug.Print sResult
        ...
End Sub
```

GetMonitorCoordinates, UIInterface Function

Syntax GetMonitorCoordinates(_nMonitor, _pLeft, _pTop, _pRight, _pBottom)

Description This function, purposely designed for multimonitor systems, can also be used in systems with one monitor only. Returns the coordinates in pixels of the area displayed by the monitor whose number is passed as parameter. The monitor index starts from value 0.

This function is only partly supported in Windows CE.(uses only the nMonitor=1 parameter, otherwise always returns 'false')

Parameter	Description
nMonitor	Number of monitor.
pLeft	Start coordinate x.
pTop	Start coordinate y.
pRight	End Coordinate x.
pBottom	End Coordinate y.

Result Boolean

Example:

```
Public Sub Click()
        ...
        Dim nMonitor As Integer
        Dim pLeft As Long
        Dim pTop As Long
        Dim pRight As Long
        Dim pBottom As Long

        nMonitor = 0
        GetMonitorCoordinates(nMonitor, pLeft, pTop, pRight, pBottom)

        Debug.Print pLeft
        Debug.Print pTop
        Debug.Print pRight
        Debug.Print pBottom
        ...
End Sub
```

GetNumMonitors, UIInterface Function

Syntax GetNumMonitors()

Description This function returns a integer value indicating the number of monitors installed in the system.

This function is not supported in Windows CE.(If set always returns 1)

Parameter	Description
None	None

Result Integer

Example:

```
Public Sub Click()
        Debug.Print GetNumMonitors
End Sub
```

GetPasswordFromLevel, UIInterface Function

Syntax GetPasswordFromLevel(_nLevel)

Description User password level request function. This function activates the Movicon user password level request specified in the parameter. If the password management has been enabled, Movicon will display the window requesting for the user and password.

Parameter	Description
nLevel As Long	User level.

Result Boolean

Example:

```
Sub Main
        ...
        'this procedure requires the password level to be enabled by means of a dialog window (see
        WinWrap Basic manual for further information on Dialog windows), 'therefore the user is
        logged on by means of password request.
        Begin Dialog UserDialog 200,150
         Text 10,10,180,15,"password level"
        OptionGroup .options
        OptionButton 10,30,180,15,"Option &1"
        OptionButton 10,45,180,15,"Option &2"
        OptionButton 10,60,180,15,"Option &3"
        OptionButton 10,75,180,15,"Option &4"
        OptionButton 10,90,180,15,"Option &5"
        OKButton 80,120,40,20
        End Dialog
        Dim level As Integer
        Dim dlg As UserDialog
        dlg.options = 0
        Dialog dlg
        level = dlg.options +1
        GetPasswordFromLevel(level)
        '...
End Sub
```

GetPasswordFromUser, UIInterface Function

Syntax GetPasswordFromUser(_lpszUser)

Description Movicon User Password request function. This function activates the specified user request (providing that the user is on the user list) in the parameter. If the password management has been enabled, Movicon will display the password request window.

Parameter	Description
lpszUser As String	User password name requested.

Result Boolean

Example:

```
Sub Main
        '...
        'This procedure requests the  "Ulrico" user password be enabled  for accessing the system
        by means of a dialog window (see the WinWrap Basic manual for further information on
        dialog windows), therefore the user is enabled through the password request.
        Dim user As String
        user=InputBox("User:","User password", "Ulrico")
        GetPasswordFromUser(User)
        ...
End Sub
```

GetVariableNameFromList, UIInterface Function

Syntax GetVariableNameFromList(_lpszVarName)

Description Allows variables to be selected from the Movicon variable database in Runtime by opening the appropriate variable Database window. This function returns a string parameter indicating the name of the variable selected by the operator and returns a boolean parameter showing True if successful or False if failed.

This function is not supported in Windows CE.(If set returns always 'false')

This function cannot be executed within a Basic Script. It can only be executed inside the screen's or the objects' code if screen has not be set as "Run in another Thread. This function can also be used in template dropipng code.

Parameter	Description
lpszVarName As String	Variable Name.

Result Boolean

Example:

```
Sub Main
        Dim sName As String
        GetVariableNameFromList (sName)
        Debug.Print sName
```

```
End Sub
```

GetWindowPos, UIInterface Function

Syntax GetWindowPos(_nX, _nY, _nWidth, _nHeight, _nShow)

Description Lets you retrieve information on the position and size of the window containing the screen.

Permits you to retrieve information on the Movicon window position and size in respect to the screen interface.

Parameter	Description
nX As Integer	top left corner X coordinate.
nY As Integer	top left corner Y coordinate.
nWidth As Integer	width size.
nHeight As Integer	height size.
nShow As Integer	show mode.

Result Boolean

Example:

```
Public Sub Click()
        Dim nX As Variant
        Dim nY As Variant
        Dim nWidth As Variant
        Dim nHeight As Variant
        Dim nShow As Variant
        GetWindowPos(nX , nY , nWidth , nHeight , nShow )
        MsgBox "nX = " & nX  &" nY = " & nY  & "nWidth = " & nWidth & "nHeight = " & nHeight &
        "nShow = " & nShow , vbOkOnly, GetProjectTitle
End Sub
```

HideLongOperationDialog, UIInterface Function

Syntax HideLongOperationDialog

Description This function closes the dialog window opened with the "ShowLongOperationDialog" function.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Main
        ...
        ShowLongOperationDialog(GetProjectTitle, True)
        ...
        HideLongOperationDialog
        …
End Sub
```

IsLongOperationAborted, UIInterface Function

Syntax IsLongOperationAborted

Description This function checks out whether a LongOperation has been aborted (with the Abort key from the ShowLongOperation Dialog Box) or not.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Main
        ...
        Dim Abort As boolean
        Abort = IsLongOperationAborted
        If Abort = 1 then ...

        HideLongOperationDialog
        ...
End Sub
```

LoadPicture, UIInterface Function

Syntax LoadPicture(_lpszFileName)

Description Returns an object containing the image specified by the lpszFileName parameter.

This function is not supported in Windows CE.(If set, always returns 'null')

Parameter	Description
lpszFileName As String	Name of image file.

Result Object

Example:

```
Public Sub Click()
        ...
        Dim imgX As ListImage
        ' Add images to ListImages collection.
        Set imgX = ImageList1.ListImages.Add(, "rocket",
        LoadPicture("D:\Documenti\Test Progetti\x_animat.gif"))
        ...
End Sub
```

LogoffActiveUser, UIInterface Function

Syntax LogoffActiveUser()

Description This function forces the logging off of the user currently logged on. This function is not available for Web Client users.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Main
        ...
        LogoffActiveUser
        ...
End Sub
```

NumericEntry, UIInterface Function

Syntax NumericEntry (_lpszVariableName, _nLowLimit, _nHighLimit)

Description Displays a numeric pad on the video screen for entering numbers such as values for the Movicon variables. This function has been purposely devised for systems without keyboards (touchscreens).

Parameter	Description
lpszVariableName As Integer	Name of the desired variable.
nLowLimit As Variant	Optional. Control value as low limit for the variable.
nHighLimit As Variant	Optional. Control value as high limit for the variable.

Result Boolean

Example:

```
Sub Main
        ...
        Dim Low As Integer, High As Integer
        Low = 10
        High = 100
        NumericEntry("VAR0001", Low, High)
        ...
End Sub
```

OpenModalSynoptic, UIInterface Function

Syntax OpenModalSynoptic(lpszSynopticName, lpszParameterFile, nX, nY, nWidth, nHeight)

Description Opens a project screen window in modal mode. A modal window is a secondary window which captures all the input by the user until it is closed.

Parameter	Description
lpszSynopticName As String	Screen window name
lpszParameterFile As String	File for screen parameters (see Screen Interface)
nX As Long	Screen window X coordinate
nY As Long	Screen Window Y coordinate
nWidth As Long	Screen window width
nHeight As Long	Screen window height

Result Boolean

Example:

```
Public Sub Click()
        ...
        OpenModalSynoptic("Sinottico1", "Param.txt", 100, 100, 600, 300)
        ...
End Sub
```

OpenSynoptic, UIInterface Function

Syntax OpenSynoptic(_lpszSynopticName, _nShow)

Description Open project screen function.

The nShow parameter is no longer considered. The screen now opens normally. This parameter is still here for previous Movicon versions compatibility so that script codes can be copied.

The "True" returned value in Windows CE indicates that the command is on hold and will be executed. For example, this means that the return value may be "True" even though the requested screen may not exist. This is due to the fact that the open screen command is executed with a delay time to allow the active screen to unload from memory (if Delay = 0) before being opened. This had been purposely done to optimize memory management.

Parameter	Description
lpszSynopticName As String	Name of screen to be opened.
nShow As Integer	Parameter not used.

Result Boolean

Example:

```
Sub Main
        ...
        OpenSynoptic("Synoptic1", 1)
        ...
End Sub
```

OpenSynopticEx, UIInterface Function

Syntax OpenSynopticEx(lpszSynopticName, nShow, lpszParameterFile, nMonitor, bNotMDIFrame, bModal)

Description Opens a project scrren window in different modes according to the specified parameters. The nMonitor indices, in multimonitor systems, which monitor the window must appear in; the bNotMDIFrame parameter enables the opening of the window outside the main frame; the bModal parameter enables the opening of the window in modal mode. The nShow NON parameter is used but for compatibility purposes with previous versions.

when a screen is opened on the secondary monitor (nMonitor > 1 parameter) it will always open in maximized mode. When a screen is opened in the same Movicon MDI frame (bNotMDIFrame = False) and not in modal (bModal parameter = False) it will always open in maximized mode. When a screen is opened in the main monitor (nMonitor = 0 parameter) and not in the same Movicon MDI frame (bNotMDIFrame = True) it will be opened with the default sizes set in the screen and with the x=0 and Y=0 coordinates.

Parameter	Description
lpszSynopticName As String	screen window name
nShow As Integer	Parameter not used
lpszParameterFile As String	file for screen's parameters
nMonitor As Integer	index number of monitor where window is to open

bNotMDIFrame As Boolean	opens the window outside the Movicon frame
bModal As Boolean	opens the window in modal mode (requires bNotMDIFrame = True)

Result Boolean

Example:

```
Public Sub Click()
        ...
        OpenSynopticEx("Sinottico1", 1, "Param.txt", 0, True, True)
        ...
End Sub
```

OpenSynopticParameter, UIInterface Function

Syntax OpenSynopticParameter(_lpszSynopticName, _nShow, _lpszParameterFile)

Description Opens the project screen specifying the parameter file in the lpszParameterFile parameter. The nShow NON parameter is used but left for compatibility purposes with previous versions. When using this command the window will always open in maximized mode.

Parameter	Description
lpszSynopticName As String	Name of screen to be opened.
nShow As Integer	Parameter not used.
lpszParameterFile As String	Parameter file.

Result Boolean

Example:

```
Sub Main
        ...
        OpenSynopticParameter("Screen1", 1, "Param.txt")
        ...
End Sub
```

SayThis, UIInterface Function

Syntax SayThis(_lpszSpeechText)

Description Creates the speech synthesis of the text string passed as parameter. This function uses Windows SAPI and requires a library of phoneme relating to the language you intend to used.

This function is not supported in Windows CE.(If set always returns 'false')

Parameter	Description
lpszSpeechText As String	Text passed to the speech synthesizer.

Result Boolean

Example:

```
Public Sub Click()
        Dim lpszText As String

        ' Select the text to speech
        lpszText = InputBox("Text to Speech:", "AgentSpeak", "I'm Robby", 100, 100)
        ' Send text to Agent
        SayThis(lpszText)
End Sub
```

SelectResourceFromList, UIInterface Function

Syntax SelectResourceFromList(_lpszTitle,_lpszResType)

Description Displays a window with the list of project resources belonging to the type singled out by the nResType parameter. The returned value indicates the name selected from the list after the window is closed with OK. When the window is closed with Cancel, the returned valued will be a null string.
This function is normally used in template codes for customizing the insertion of symbols from the library. For further information on this, please consult the help of the OnCustomizeSymbol event.
resource type:

enum_SYNOPTIC_RESOURCES = Screens.
enum_SCRIPT_RESOURCES = Basic Scripts.
enum_ACCELERATOR_RESOURCES = Accelerators.
enum_MENU_RESOURCES = Menus.
enum_PARAMETER_RESOURCES =Parameter File
enum_REPORT_RESOURCES (valore 32) = Embedded Report

This function is not supported in Windows CE.(If set, always returns an empty string)

This function cannot be executed within Basic Script resources. It can only be executed in object and screen code and in template dropping code.

Parameter	Description
lpszTitle As String	Window title.
lpszResType As Integer	Resource type. More resource types can be linked, for example by writing: enum_SYNOPTIC_RESOURCES Or enum_SCRIPT_RESOURCES in this way the resource selection window in will open with two tabs, one for selecting screens and the other one for selecting basic script.

Result String

Example:

```
Sub OnCustomizeSymbol (...)
        Dim sSyn As String
        sSyn = SelectResourceFromList ("", enum_SYNOPTIC_RESOURCES Or
        enum_SCRIPT_RESOURCES)
        If sSyn = "" Then
        bRet = False
        Else
        Prop("Syn") = sSyn
        bShowPropInsp = False
        End If
End Sub
```

SetDefPrinterOrient, UIInterface Function

Syntax SetDefPrinterOrient(_bLandScape)

Description This function allows you to set the predefined system printer orient. When set at True the orient will be horizontal, when set at False the orient will be vertical.

This property is not supported in Windows CE.

Parameter	Description
bLandScape As Boolean	Printer's horizontal orient.

Result None

Example:

```
Public Sub Click()
        ...
        SetDefPrinterOrient(True)
        ...
End Sub
```

SetRedraw, UIInterface Function

Syntax SetRedraw(_bSet)

Description This function permits you to enable or disable the graphics of opened screens. The bSet parameter identifies one of the two modes.
This function when used for disabling, may cause the lose of user interactivity with the other objects or symbols on the screen page. This function is useful for disabling the displaying of long graphical operations in the project to activate them only when they have been completed.

Parameter	Description

bSet As Boolean	Enable value.

Result None

Example:

```
Sub Main
        If MsgBox("Do you suspend redraw on the synoptic ?", vbYesNo, "SetRedraw") = vbYes
        Then
         SetRedraw(False)
         Wait 5
        End If
        SetRedraw(True)
End Sub
```

SetWindowPos, UIInterface Function

Syntax SetWindowPos(_nX, _nY, _nWidth, _nHeight, _nShow)

Description This function permits you to move and resize the Movicon window.

The nShow parameter can obtain the following values:

0= hide
1= restore
2= reduce to icon
3= maximize

Parameter	Description
nX As Integer	top left corner X coordinate.
nY As Integer	top left corner Y coordinate.
nWidth As Integer	width size.
nHeight As Integer	height size.
nShow As Integer	Display mode definitions: 0= hide 1= restore 2= reduce to icon 3= maximize

Result Boolean

Example:

```
Sub Main
        ...
        SetWindowPos(100,100,450,450,1)
        ...
End Sub
```

ShowHTMLDialog, UIInterface Function

Syntax ShowHTMLDialog(_lpszURL, _varArgIn, _varArgOut)

Description Shows a dialog window containing the HTML page specified in the lpszURL parameter. You can get and pass values of any type by using the varArgIn and varArgOut parameters.
The dialog window which opens is in modal and therefore other external commands cannot be accessed until it is closed.
The return value indicates whether the dialog window has opened successfully (True) or not (False).

This function is not supported in Windows CE. In this context the function will not execute any command and returns the "False" value.

The two varArgIn and varArgOut parameters must be managed in the html page using the window.dialogArguments and window.returnValue functions.
For further information please also refer to the online documention about using DHTML, Java Script, etc.
(http://msdn.microsoft.com/en-us/library/ms533723(VS.85).aspx
http://msdn.microsoft.com/en-us/library/ms534371.aspx)

The "ES_ShowHTMLDialog" example is also available on the Progea website.

Parameter	Description
lpszURL As String	Path and name (including extension) of the HTML page to be opened.
varArgIn As Variant	Optional: Input values passed to HTML page
varArgOut As Variant	Optional: Output values recieved from the HTML page.

Result Boolean

Example:

```
Sub Main
        ...
        Dim lpszURL As String
        Dim varArgIn As Variant
        Dim varArgOut As Variant

        lpszURL = "C:\Documenti\Index.html"
        varArgIn = 5
        ShowHTMLDialog(lpszURL, varArgIn, varArgOut)
        MsgBox ("varArgIn=" & varArgIn & vbLf & "varArgOut=" & varArgOut, "ShowHTMLDialog")
        ...
End Sub
```

ShowLongOperationDialog, UIInterface Function

Syntax ShowLongOperationDialog(_lpszTitle, _bCancelOperation)

Description This function shows a modal dialog window through which a message can be displayed to warn the user that a certain operation is underway. The "bCancelOperation" parameter, when set at True, allows the Cancel key to be displayed in the window so that the window can be closed before being done by using the "HideLongOperationDialog" function. You will need to keep in mind that when pressing down the Cancel key on the ShowLongOperationDialog an internal memory is activated allowing you to test with the "IsLongOperationAborted" function if the ShowLongOperationDialog has been cancelled. The ShowLongOperationDialog will not be displayed for this routine (basic script resources, drawings, etc...) until the memory is cleared. The memory is reset by using the "HideLongOperationDialog" method.

Parameter	Description
lpszTitle As String	Window title.
bCancelOperation As Boolean	'Cancel' key presence for closing window.

Result Boolean

Example:

```
Sub Main
    ...
    ShowLongOperationDialog("Movicon", True)
    ...
    HideLongOperationDialog
End Sub
```

ShowMenu, UIInterface Function

Syntax ShowMenu(_lpszMenu)

Description This allows a menu resource from the project's resources to be showed. The menu window will be displayed in correspondence to the mouse pointer's position.

Parameter	Description
lpszMenu As String	Menu to be displayed.

Result Boolean

Example:

```
Sub Main
    ...
    ShowMenu("Menu1")
    ...
End Sub
```

ActiveLanguage, UIInterface Property

Syntax ActiveLanguage = _String

Description Permits the active language to be set or read.

Parameter	Description
None	None

Result String

Example:

```
Public Sub Click()
        Dim sLanguage As String
        sLanguage = ActiveLanguage
        Debug.Print sLanguage
End Sub
```

MainVisible, UIInterface Property

Syntax MainVisible = _Boolean

Description When set to boolean value 0, the Movicon application window will be made invisible. When set back to value 1, the Movicon will be restored to visibility.

Parameter	Description
None	None

Result Boolean

Example:

```
Sub Main
        Debug.Print "MainVisible: " & MainVisible
        MainVisible = False
        Wait 5
        MainVisible = True
End Sub
```

56. UserAndGroupCmdTarget

GetActiveUserObject, UserAndGroupCmdTarget Function

Syntax GetActiveUserObject()

Description This function permits you to get the user object relating to the user currently logged on. Once the object has been retrieved, all the methods and properties described in the UserCmdTarget interface will be made available. When no user has logged on, this function will return a Nothing object.

Parameter	Description
None	None

Result Object
If Function has been executed successfully it will retrieve an object of type UserCmdTarget if otherwise Nothing is returned.

Example:

```
Option Explicit
Public Sub Click()
        Dim objUserAndGroup As UserAndGroupCmdTarget
        Dim objUser As UserCmdTarget

        Set objUserAndGroup = GetUserAndGroup
        If Not objUserAndGroup Is Nothing Then
                Set objUser = objUserAndGroup.GetActiveUserObject
                If Not objUser Is Nothing Then
                        MsgBox(objUser.Name & " is Logged In", vbOkOnly,
                        GetProjectTitle)
                        Set objUser = Nothing
                End If
                Set objUserAndGroup = Nothing
        End If
End Sub
```

GetDesignGroupAtPos, UserAndGroupCmdTarget Function

Syntax GetDesignGroupAtPos(nPosition)

Description This function allows you to retrieve the "User group" object, defined in the list of user groups during design mode, identified by the parameter's nPosition. Once this object has been retrieved, all the methods and properties from the UserGroupCmdTarget interface will be made available.
The retrieved object is different from Nothing only when a user has logged in (i.e. by using the 'LogonUser' function or user logging in from consol) with a level equal or higher to the administration level. Like "GetGroupObject()", this function is protected against access by lower level users.

Parameter	Description
nPosition As Integer	Group's reference index is the same in the list of user groups defined in design mode.

Result Object
A UserGroupCmdTarget type object is returned if the function has been executed successfully, otherwise object returns Nothing.

Example:

```
Option Explicit
Public Sub Click()

        Dim objUserAndGroup As UserAndGroupCmdTarget
        Dim objGroup As UserGroupCmdTarget
        Dim nNumDesignGroups As Integer

        Set objUserAndGroup = GetUserAndGroup
        If Not objUserAndGroup Is Nothing Then
                objUserAndGroup.LogonUser("Administrator", "Administrator")
        End If

        nNumDesignGroups= objUserAndGroup.GetNumDesignGroups()
        MsgBox "Number of design groups = " & nNumDesignGroups

        For i = 0 To nNumDesignGroups - 1
                Set objGroup = objUserAndGroup.GetDesignGroupAtPos(i)
                If Not objGroup Is Nothing Then
                        MsgBox "Design group at pos " & i & " is: " & objGroup.Name
                End if
        Next
End Sub
```

GetGroupObject, UserAndGroupCmdTarget Function

Syntax GetGroupObject(_lpszGroupName)

Description This function permits you to get the user group object referred to by the lpszGroupName parameter. Once the object has been retrieved all the methods and properties of the UserGroupCmdTarget interface will be made available.
The returned object is different from Nothing only when the log in of a user has been carried out (by either using the 'LogonUser' function or logging in from a consol) with a level equal to or higher than the administrator level. This function, like the "GetUserObject()", is protected against those with lower levels.

The GetGroupObject function consents you to also retrieve the reference of a group created in runtime. The group is first search for in the list of those configured in the project and then in those created in runtime. As a consequence if there is a group with the same name configured in design and in runtime mode, the one configured in design mode will be the one retrieved.

Parameter	Description
lpszGroupName As String	Name of Group to be retrieved.

Result Object
If Function has been executed successfully it will retrieve an object of type UserGroupCmdTarget if otherwise Nothing is returned.

Example:

```
Option Explicit
Public Sub Click()
        Dim obj1 As UserAndGroupCmdTarget
        Dim obj2 As UserGroupCmdTarget
        Dim obj3 As UserCmdTarget
        Set obj1 = GetUserAndGroup
        If Not obj1 Is Nothing Then
                Set obj2 = obj1.GetGroupObject("Group1")
                Set obj3 = obj1.GetUserObject("Group1","Ut1")
                Debug.Print obj1.EnableAutoLogoff
                Set obj1 = Nothing
                 If Not obj2 Is Nothing Then
                        Debug.Print obj2.Description
                        Set obj2 = Nothing
                 End If
                 If Not obj3 Is Nothing Then
                        Debug.Print obj3.Description
                        Set obj3 = Nothing
                 End If
        End If
End Sub
```

GetNumDesignGroups, UserAndGroupCmdTarget Function

Syntax GetNumDesignGroups()

Description This function returns the number of user groups defined in design mode.

Parameter	Description
None	None

Result Long

Example:

```
Option Explicit
Public Sub Click()

Dim objUserAndGroup As UserAndGroupCmdTarget
Dim objGroup As UserGroupCmdTarget
Dim nNumDesignGroups As Integer

        Set objUserAndGroup = GetUserAndGroup
        If Not objUserAndGroup Is Nothing Then
                objUserAndGroup.LogonUser("Administrator", "Administrator")
        End If

        nNumDesignGroups= objUserAndGroup.GetNumDesignGroups()
        MsgBox "Number of design groups = " & nNumDesignGroups
        For i = 0 To nNumDesignGroups - 1
```

```
                Set objGroup = objUserAndGroup.GetDesignGroupAtPos(i)
                If Not objGroup Is Nothing Then
                        MsgBox "Design group at pos " & i & " is: " & objGroup.Name
                End if
        Next
End Sub
```

GetNumActiveUsers, UserAndGroupCmdTarget Function

Syntax GetNumActiveUsers()

Description This function returns the number of users currently logged on.

Parameter	Description
None	None

Result Long

Example:

```
Option Explicit
Public Sub Click()
        Dim myObject As UserAndGroupCmdTarget
        Dim numUsers As Long
        Set myObject = GetUserAndGroup
        If Not myObject Is Nothing Then
                numUsers = myObject .GetNumActiveUsers
                MsgBox(numUsers & " Users are Logged In", vbOkOnly,
                GetProjectTitle)
        End If
End Sub
```

GetNumRuntimeGroups, UserAndGroupCmdTarget Function

Syntax GetNumRuntimeGroups()

Description This function returns the number of user groups defined in runtime using the "EditUser" tool.

Parameter	Description
None	None

Result Long

Example:

```
Option Explicit
Public Sub Click()
Dim objUserAndGroup As UserAndGroupCmdTarget
Dim objGroup As UserGroupCmdTarget
Dim nNumRuntimeGroups As Integer

Set objUserAndGroup = GetUserAndGroup

        If Not objUserAndGroup Is Nothing Then
                objUserAndGroup.LogonUser("Administrator", "Administrator")
        End If
        nNumRuntimeGroups= objUserAndGroup.GetNumRuntimeGroups()
        MsgBox "Number of Runtime groups = " & nNumRuntimeGroups
        For i = 0 To nNumRuntimeGroups - 1
                Set objGroup = objUserAndGroup.GetRuntimeGroupAtPos(i)
                If Not objGroup Is Nothing Then
                        MsgBox "Runtime group at pos " & i & " is: " & objGroup.Name
                End if
        Next
End Sub
```

GetRuntimeGroupAtPos, UserAndGroupCmdTarget Function

Syntax GetRuntimeGroupAtPos(nPosition)

Description This function allows you to retrieve the "User group" object, defined in the list of user groups during runtime mode, identified by the parameter's nPosition. Once this object has been retrieved, all the methods and properties from the UserGroupCmdTarget interface will be made available.
The retrieved object is different from Nothing only when a user has logged in (i.e. by using the 'LogonUser' function or user logging in from consol) with a level equal or higher to the administration level. Like "GetGroupObject()", this function is protected against access by lower level users.

Parameter	Description
nPosition As Integer	Group's reference index is the same in the list of user groups defined in runtime mode.

Result Object
A UserGroupCmdTarget type object is returned if the function has been executed successfully, otherwise object returns Nothing.

Example:

```
Option Explicit
Public Sub Click()
Dim objUserAndGroup As UserAndGroupCmdTarget
Dim objGroup As UserGroupCmdTarget
Dim nNumRuntimeGroups As Integer

        Set objUserAndGroup = GetUserAndGroup
        If Not objUserAndGroup Is Nothing Then
                objUserAndGroup.LogonUser("Administrator", "Administrator")
        End If

        nNumRuntimeGroups= objUserAndGroup.GetNumRuntimeGroups()
        MsgBox "Number of Runtime groups = " & nNumRuntimeGroups

        For i = 0 To nNumRuntimeGroups - 1
```

```
                Set objGroup = objUserAndGroup.GetRuntimeGroupAtPos(i)
                If Not objGroup Is Nothing Then
                        MsgBox "Runtime group at pos " & i & " is: " & objGroup.Name
                End if
        Next
End Sub
```

GetUserObject, UserAndGroupCmdTarget Function

Syntax GetUserObject(_lpszGroupName, _lpszUserName)

Description This function permits you to get the user object referenced by the lpszUserName parameter belonging to the lpszGroupName group. Once this object has been retrieved, all the methods and properties described in the UserCmdTarget interface will be made available. The returned object is different from Nothing only when a user has logged on (either by using the 'LogonUser' function or logging on from a console) with a level equal to or higher than administrator. This function, like the "GetGroupObject()", is protected against lower access levels.

Parameter	Description
lpszGroupName As String lpszUserName As String	Name of group which user object belongs to. Name of user to be fetched.

Result Object
If Function has been executed successfully it will retrieve an object of type UserCmdTarget if otherwise Nothing is returned.

Example:

```
Option Explicit
Public Sub Click()
        Dim obj1 As UserAndGroupCmdTarget
        Dim obj2 As UserGroupCmdTarget
        Dim obj3 As UserCmdTarget
        Set obj1 = GetUserAndGroup
        If Not obj1 Is Nothing Then
                Set obj2 = obj1.GetGroupObject("Group1")
                Set obj3 = obj1.GetUserObject("Group1","Ut1")
                Debug.Print obj1.EnableAutoLogoff
                Set obj1 = Nothing
                 If Not obj2 Is Nothing Then
                        Debug.Print obj2.Description
                        Set obj2 = Nothing
                 End If
                 If Not obj3 Is Nothing Then
                        Debug.Print obj3.Description
                        Set obj3 = Nothing
                 End If
        End If
End Sub
```

GetXMLSettings, UserAndGroupCmdTarget Function

Syntax GetXMLSettings

Description This function returns a string with the XML file content of the project relating to the Users and User Groups.

Note: The function and the properties for managing Groups and Users can be accessed only if a user has logged on (by using the 'LogonUser' function or by user logging in from console) with a level equal or higher than administrator level.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim MyUsersAndGroups As UserAndGroupCmdTarget
        GetPasswordFromLevel(0)
        Set MyUsersAndGroups = GetUserAndGroup
        If Not MyUsersAndGroups Is Nothing Then
                MsgBox MyUsersAndGroups.GetXMLSettings,vbOkOnly,""
        End If
        Set MyUsersAndGroups = Nothing
End Sub
```

LogonUser, UserAndGroupCmdTarget Function

Syntax LogonUser(_lpszUserName,_lpszPassword)

Description This function allows you to execute the logon for the user referred by the user name and by the password passed as parameters.

Parameter	Description
lpszUserName As String	User Name
lpszPassword As String	Password

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
        Dim myObject As UserAndGroupCmdTarget
        Set myObject = GetUserAndGroup
        If Not myObject Is Nothing Then
```

```
                If myObject.LogonUser("Utente1","pwdUtente1") then
                        MsgBox("Utente1 has logged in!", vbOkOnly, GetProjectTitle)
                End If
        End If
End Sub
```

ReadRuntimeUsersXML, UserAndGroupCmdTarget Function

Syntax ReadRuntimeUsersXML(vUserName, vPassword)

Description This function returns a string with the project file's XML contents relating to the runtime users list.
Note: The parameters relating to the user name and password must correspond to a user level equal or higher than administrator level. In cases where these parameters are not set, a user with a level equal to or higher than administrator need to be logged on by means of using the "LogonUser" function or from consol)

Parameter	Description
vUserName As String	Name of user level equal or higher than Administrator level (default = "").
vPassword As Strings	Password of user specified in the "vUserName" parameter (default = "").

Result String

Example:

```
Option Explicit
Sub Main
        Dim sXML As String
        Dim sUserName As String
        Dim sPassword As String
        sUserName = This.GetParameter(0)
        sPassword = This.GetParameter(1)
        If sUserName <> "" Then
                sXML = GetUserAndGroup.ReadRuntimeUsersXML(sUserName,
                sPassword)
        Else
                GetUserAndGroup.LogonUser("Progea", "Progea")
                sXML = GetUserAndGroup.ReadRuntimeUsersXML()
                LogoffActiveUser
        End If
End Sub
```

SaveRuntimeUsersXML, UserAndGroupCmdTarget Function

Syntax SaveRuntimeUsersXML(lpszNewXMLSettings, bCrypt, vUserName, vPassword)

Description This function saves the project file's XML contents relating to the runtime users list.
Note: The parameters relating to the user name and password must correspond to a user level equal or higher than administrator level. In cases where these parameters are not set, a user with a level equal to or higher than administrator

need to be logged on by means of using the "LogonUser" function or from consol)

Parameter	Description
lpszNewXMLSettings As String	XML String containing runtime users: <UserAndGroupSettings> ... ... </UserAndGroupSettings>
bCrypt As Boolean	lets choose where to save runtime users file in crypted mod or not. bCrypt = True: The runtme users file is saved in crypted mode and therefore cannot not be read by opening a text editor. bCrypt = False: The runtime users file is not saved in crypted mode and therefore can be read by opening a text editor.
vUserName As String	Name of user with level equal to or higher than Administrator (default = "").
vPassword As Strings	Password of user specified in the "'vUserName" parameter (default = "").

Result Boolean

Example:

```
Option Explicit

Sub Main
        Dim sXML As String
        Dim sUserName As String
        Dim sPassword As String
        Dim bCrypt As Boolean
        Dim bResult As Boolean

        sUserName = This.GetParameter(0)
        sPassword = This.GetParameter(1)
        bCrypt = CBool(Val(This.GetParameter(2)))

        If sUserName <> "" Then
                sXML = GetUserAndGroup.ReadRuntimeUsersXML(sUserName,
                sPassword)
                bResult = GetUserAndGroup.SaveRuntimeUsersXML(sXML, bCrypt,
                sUserName, sPassword)
        Else
                GetUserAndGroup.LogonUser("Progea", "Progea")
                sXML = GetUserAndGroup.ReadRuntimeUsersXML
                bResult = GetUserAndGroup.SaveRuntimeUsersXML(sXML, bCrypt)
                LogoffActiveUser
        End If

        If bResult = False Then
                MsgBox "Error on writing the runtime xml users file!", vbExclamation +
                vbOkOnly, GetProjectTitle
        End If
End Sub
```

AllowResizingForUsersBelowThisLevel, UserAndGroupCmdTarget Property

Syntax AllowResizingForUsersBelowThisLevel = _Integer

Description This property is used for reading or setting the user level required for system operations to resize or reduce the Movicon window to an icon. When setting this level to "0", no user level authentication will be requested for resizing or reducing the Movicon Window to an icon. However, any attempt to close this window will need Administration level (1023) authorization independently from the value set in this property.
This property is read or written only if the user has logged in with a level equal or higher than administrator level (for example, by using the "LogonUser" function or the user login from consol).

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
        Dim oUserGroup As UserAndGroupCmdTarget

        Set oUserGroup = GetUserAndGroup()
        oUserGroup.LogonUser("Administrator", "Administrator")
        oUserGroup.AllowResizingForUsersBelowThisLevel = 0
        Set oUserGroup = Nothing
End Sub
```

AllowRuntimeChangesForUsersBelowThisLevel, UserAndGroupCmdTarget Property

Syntax AllowRuntimeChangesForUsersBelowThisLevel = _Integer

Description By using this property you can change the Users Level above which it will not be possible to make any changes in Runtime.
For instance, if you set "User Level 5 (5)", only users below this level can make changes during Runtime.

The values for the various levels are:

```
enum_UserLevel0 = level 0
enum_UserLevel1 = level 1
enum_UserLevel2 = level 2
enum_UserLevel3 = level 3
enum_UserLevel4 = level 4
enum_UserLevel5 = level 5
enum_UserLevel6 = level 6
enum_UserLevel7 = level 7
enum_UserLevel8 = level 8
enum_UserLevelAdministrator     = administrator level
enum_UserLevelDeveloper = Developer Level
```

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
        Dim myObject As UserAndGroupCmdTarget
        Set myObject = GetUserAndGroup
        If Not myObject Is Nothing Then
                myObject.AllowRuntimeChangesForUsersBelowThisLevel = enum_UserLevel6
        End If
End Sub
```

DefaultPrivAdminAccessLevel, UserAndGroupCmdTarget Property

Syntax DefaultPrivAdminAccessLevel = _Long

Description This property sets or returns the Access Levels to be assigned to the Operating System users belonging to the Administators group.

This property is not supported in Windows CE.(Always returns zero if set)

Parameter	Description
None	None

Result Long

Example:

```
Option Explicit
Public Sub Click()
        Dim myObject As UserAndGroupCmdTarget
        Set myObject = GetUserAndGroup
        If Not myObject Is Nothing Then
                Debug.Print myObject.DefaultPrivAdminAccessLevel
        End If
End Sub
```

DefaultPrivAdminLevel, UserAndGroupCmdTarget Property

Syntax DefaultPrivAdminLevel = _Integer

Description This property sets or returns the default Password level to be assigned to the Operation System Users belonging to the Administrators group.

The values for the levels are:

enum_UserLevel0 = level 0
enum_UserLevel1 = level 1
enum_UserLevel2 = level 2
enum_UserLevel3 = level 3
enum_UserLevel4 = level 4
enum_UserLevel5 = level 5
enum_UserLevel6 = level 6
enum_UserLevel7 = level 7
enum_UserLevel8 = level 8
enum_UserLevelAdministrator = Administrator level
enum_UserLevelDeveloper = Developer level

This property is not supported in Windows CE.(Always returns zero if set)

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
        Dim myObject As UserAndGroupCmdTarget
        Set myObject = GetUserAndGroup
        If Not myObject Is Nothing Then
                myObject.DefaultPrivAdminLevel = enum_UserLevelAdministrator
        End If
End Sub
```

DefaultPrivGuestAccessLevel, UserAndGroupCmdTarget Property

Syntax DefaultPrivGuestAccessLevel = _Long

Description This property sets or returns the Access Levels to be assigned to the Operating System Users belonging to the Guests group.

This property is not supported in Windows CE.(Always returns zero if set)

Parameter	Description
None	None

Result Long

Example:

```
Option Explicit
Public Sub Click()
        Dim myObject As UserAndGroupCmdTarget
        Set myObject = GetUserAndGroup
        If Not myObject Is Nothing Then
                Debug.Print myObject.DefaultPrivGuestAccessLevel
        End If
End Sub
```

DefaultPrivGuestLevel, UserAndGroupCmdTarget Property

Syntax DefaultPrivGuestLevel = _Integer

Description This property sets or returns the default Password level to be assigned to the Operation System Users belonging to the Guests group.

The values for the levels are:

enum_UserLevel0 = level 0
enum_UserLevel1 = level 1
enum_UserLevel2 = level 2
enum_UserLevel3 = level 3
enum_UserLevel4 = level 4
enum_UserLevel5 = level 5
enum_UserLevel6 = level 6
enum_UserLevel7 = level 7
enum_UserLevel8 = level 8
enum_UserLevelAdministrator = Administrator level
enum_UserLevelDeveloper = Developer level

This property is not supported in Windows CE.(Always returns zero if set)

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
        Dim myObject As UserAndGroupCmdTarget
        Set myObject = GetUserAndGroup
        If Not myObject Is Nothing Then
                myObject.DefaultPrivGuestLevel = enum_enum_UserLevel0
        End If
End Sub
```

DefaultPrivUserAccessLevel, UserAndGroupCmdTarget Property

Syntax DefaultPrivUserAccessLevel = _Long

Description This property sets or returns the Access Levels to be assigned to the Operating System Users belonging to the Users Group.

This property is not supported in Windows CE.(Always returns zero if set)

Parameter	Description
None	None

Result Long

Example:

```
Option Explicit
Public Sub Click()
        Dim myObject As UserAndGroupCmdTarget
        Set myObject = GetUserAndGroup
        If Not myObject Is Nothing Then
                Debug.Print myObject.DefaultPrivUserAccessLevel
        End If
End Sub
```

DefaultPrivUserLevel, UserAndGroupCmdTarget Property

Syntax DefaultPrivUserLevel = _Integer

Description This property sets or returns the default Password level to be assigned to the Operation System Users belonging to the Users group.

The values for the levels are:

enum_UserLevel0 = level 0
enum_UserLevel1 = level 1
enum_UserLevel2 = level 2
enum_UserLevel3 = level 3
enum_UserLevel4 = level 4
enum_UserLevel5 = level 5
enum_UserLevel6 = level 6
enum_UserLevel7 = level 7
enum_UserLevel8 = level 8
enum_UserLevelAdministrator = Administrator level
enum_UserLevelDeveloper = Developer level

This property is not supported in Windows CE.(Always returns 'false' if set)

Parameter	Description

None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
	Dim myObject As UserAndGroupCmdTarget
	Set myObject = GetUserAndGroup
	If Not myObject Is Nothing Then
		myObject.DefaultPrivUserLevel = enum_enum_UserLevel5
	End If
End Sub
```

EnableAutoLogoff, UserAndGroupCmdTarget Property

Syntax EnableAutoLogoff = _Boolean

Description This property sets or returns the AutoLogoff configuration for Users and User Groups. When set with a True boolean value, the Logoff will be executed automatically by Movicon after the time set in the "Logoff Automatico dopo (sec)" property. The time count will start the moment the user stops using the mouse or keyboard for executing operations.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
	Dim myObject As UserAndGroupCmdTarget
	Set myObject = GetUserAndGroup
	If Not myObject Is Nothing Then
		MsgBox("EnableAutoLogoff is: " & CBool(myObject.EnableAutoLogoff),vbOkOnly,GetProjectTitle)
	End If
End Sub
```

EnableNTUserLogin, UserAndGroupCmdTarget Property

Syntax EnableNTUserLogin = _Boolean

Description This property sets or returns the 'Enable Windows Users' configuration.

This property is not supported in Windows CE.(if used, always returns 'false')

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
	Dim myObject As UserAndGroupCmdTarget
	Set myObject = GetUserAndGroup
	If Not myObject Is Nothing Then
		MsgBox("EnableNTUserLogin is: " & CBool(myObject.EnableNTUserLogin),vbOkOnly,GetProjectTitle)
	End If
End Sub
```

EnableRuntimeUsers, UserAndGroupCmdTarget Property

Syntax EnableRuntimeUsers = _Boolean

Description This property, when set with the True boolean value, permits the Runtime Users management to be activated so that new users can be added, changed or deleted during project Runtime.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
	Dim myObject As UserAndGroupCmdTarget
	Set myObject = GetUserAndGroup
	If Not myObject Is Nothing Then
		MsgBox("EnableRuntimeUsers is: " & CBool(myObject.EnableRuntimeUsers),vbOkOnly,GetProjectTitle)
	End If
End Sub
```

MinimumPasswordLength, UserAndGroupCmdTarget Property

Syntax MinimumPasswordLength = _Long

Description This property sets or returns the minimum number of characters to be used for the Password of the set project Users.

The minimum User password length set for default by Movicon is six characters. It is advised not to use less characters than six due to normative and security reasons.

Parameter	Description
None	None

Result Long

Example:

```
Option Explicit
Public Sub Click()
        Dim myObject As UserAndGroupCmdTarget
        Set myObject = GetUserAndGroup
        If Not myObject Is Nothing Then
                myObject.MinimumPasswordLength = 60
        End If
End Sub
```

MinimumUserLength, UserAndGroupCmdTarget Property

Syntax MinimumUserLength = _Long

Description This property sets or returns the minimum number of characters to be used for the Names of users setup in the project.

The minimum User Name length is four characters set for default by Movicon. It is advised not to use less than four characters due to normative and security reasons.

Parameter	Description
None	None

Result Long

Example:

```
Option Explicit
Public Sub Click()
        Dim myObject As UserAndGroupCmdTarget
        Set myObject = GetUserAndGroup
        If Not myObject Is Nothing Then
                myObject.MinimumUserLength = 6
```

```
        End If
End Sub
```

SecsAutoLogoffTimeout, UserAndGroupCmdTarget Property

Syntax SecsAutoLogoffTimeout = _Long

Description This propery sets or returns the time in seconds of the autologoff after which the Logoff will be executed automatically by Movicon if the EnableAutoLogoff property has been set with the True boolean value.

Parameter	Description
None	None

Result Long

Example:

```
Option Explicit
Public Sub Click()
        Dim myObject As UserAndGroupCmdTarget
        Set myObject = GetUserAndGroup
        If Not myObject Is Nothing Then
                myObject.SecsAutoLogoffTimeout= 35
        End If
End Sub
```

57. UserCmdTarget

GetDaysBeforePasswExpires, UserCmdTarget Function

Syntax GetDaysBeforePasswExpires()

Description This function returns the number of days that have gone by since the user was created or since the expired password was renewed. Once expired, the User will be asked to insert a new password at the next authentication. This calculation is based on the value set in the Expiring Password (days) or by using a basic code with the ExpiringDaysPassword property. When the value of this property is left at zero, the value set in the "Expiring Password (Days)" property of the Group it belongs to.
This functioin calculates the days based on the "TimePassword" attribute in the .movprj file for Developer users and in the .rtusers file for Runtime users. It gets populated first with user creation date and time, example:

TimePassword="2009-03-13T10:19:42Z"

The function can be accessed only when a user is logged on (by either using the 'LogonUser' function or logging in from a consol) with a level equal to or higher than the administrator level and when the reference user has a User Level lower than the value set by AllowRuntimeChangesForUsersBelowThisLevel Property.
Otherwise this function will return a value equal to 0.

Parameter	Description
None	None

Result Long

Example:

```
Option Explicit
Public Sub Click()
        Dim obj1 As UserAndGroupCmdTarget
        Dim obj2 As UserCmdTarget
        Set obj1 = GetUserAndGroup
        If Not obj1 Is Nothing Then
                Set obj2 = obj1.GetUserObject("Group1","Ut1")
                Set obj1 = Nothing
                If Not obj2 Is Nothing Then
                        Debug.Print obj2.GetDaysBeforePasswExpires
                        Set obj2 = Nothing
                End If
         End If
End Sub
```

GetGroupObject, UserCmdTarget Function

Syntax GetGroupObject()

Description This function returns an UserGroupCmdTarget object type relating to the group the user belongs to. This function can be used to access the properties and methods relating to the Group.
The returned object is different from Nothing only when the log in of a user has been carried out (by either using the 'LogonUser' function or logging in from a consol) with a level equal to or higher than the administrator level. This function is protected against those with lower levels.

Parameter	Description
None	None

Result Object
If Function has been executed successfully it will retrieve an object of type UserGroupCmdTarget if otherwise Nothing is returned.

Example:

```
Option Explicit
Public Sub Click()
        Dim obj1 As UserAndGroupCmdTarget
        Dim obj2 As UserCmdTarget
        Dim obj3 As UserGroupCmdTarget

        Set obj1 = GetUserAndGroup
        If Not obj1 Is Nothing Then
                Set obj2 = obj1.GetUserObject("Group1","Ut1")
                Set obj1 = Nothing
                If Not obj2 Is Nothing Then
                        Set obj3 = obj2.GetGroupObject
                        Set obj2 = Nothing
                        If Not obj3 Is Nothing Then
                                Debug.Print obj3.Name
                                Set obj3 = Nothing
                        End If
                End If
        End If
End Sub
```

GetLastTimeUserAction, UserCmdTarget Function

Syntax GetLastTimeUserAction()

Description This function returns the last time the user interacted with the system.
This function can be accessed only when a user is logged on (by either using the 'LogonUser' or the by logging on from a console) with a level equal to or higher than the administrator level and when the reference use has a User Level lover than the set by the AllowRuntimeChangesForUsersBelowThisLevel property.
Otherwise the function will return a value equal to 0.00.00.

Parameter	Description

None	None

Result Date

Example:

```
Option Explicit
Sub Main()
	Dim obj1 As UserAndGroupCmdTarget
	Dim obj2 As UserCmdTarget
	Set obj1 = GetUserAndGroup
	If Not obj1 Is Nothing Then
		Set obj2 = obj1.GetUserObject("Group1","Ut1")
		Set obj1 = Nothing
		If Not obj2 Is Nothing Then
			Debug.Print obj2.GetLastTimeUserAction
			Set obj2 = Nothing
		End If
	End If
End Sub
```

GetListAccessVariable, UserCmdTarget Function

Syntax GetListAccessVariable()

Description This function returns a string containing the list of variables which can be accessed by the user based on their user User Access Level. To refresh this list, in case where changes have been made in their configurations, please refer to the ResetListAccessVariables function.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
	Dim obj1 As UserAndGroupCmdTarget
	Dim obj2 As UserCmdTarget
	Set obj1 = GetUserAndGroup
	If Not obj1 Is Nothing Then
		Set obj2 = obj1.GetActiveUserObject
		Set obj1 = Nothing
		If Not obj2 Is Nothing Then
			obj2.ResetListAccessVariables
			Debug.Print obj2.GetListAccessVariable
			Set obj2 = Nothing
		End If
	End If
End Sub
```

GetProp, UserCmdTarget Function

Syntax GetProp(_lpszPropID)

Description This function allows you to retrieve a previously set variable value for the selected user. In this way properties can be managed, for example from a web client, which are maintained for the total duration of session which that user is operating in. These properties are not retentive and therefore are lost when the user logs off.

Parameter	Description
lpszPropID	ID name of the property to be retrieved.

Result String

Example 1:

```
'Code executable also on WebClient
Option Explicit
Public Sub Click()
        Dim objUser As UserCmdTarget

        Set objUser = GetSynopticObject.GetActiveUserObject
        If Not objUser Is Nothing Then
                objUser.SetProp("Prop1","Name")
                objUser.SetProp("Prop2","Description")
                Debug.Print "Prop1 = " & objUser.GetProp("Prop1") & "  -  " & "Prop2 =
                " & objUser.GetProp("Prop2")
                Set objUser = Nothing
        End If
End Sub
```

Example 2:

```
'Code not executable on WebClient
Option Explicit
Public Sub Click()
        Dim objUserAndGroup As UserAndGroupCmdTarget
        Dim objUser As UserCmdTarget

        Set objUserAndGroup = GetUserAndGroup
        If Not objUserAndGroup Is Nothing Then
                Set objUser = objUserAndGroup.GetActiveUserObject
                If Not objUser Is Nothing Then
                        objUser.SetProp("Prop1","Name")
                        objUser.SetProp("Prop2","Description")
                        Debug.Print "Prop1 = " & objUser.GetProp("Prop1") & "  -  " &
                        "Prop2 = " & objUser.GetProp("Prop2")
                        Set objUser = Nothing
                End If
                Set objUserAndGroup = Nothing
        End If
End Sub
```

GetXMLSettings, UserCmdTarget Function

Syntax GetXMLSettings()

Description This function returns a string with the contents of the project's XML file relating to the user.
This function can only be accessed when a user has logged on (either by using the 'LogonUser' function or logging on from a console) with a level equal to or higher than administrator and when the referenced user has a User Level lower than the value set in the AllowRuntimeChangesForUsersBelowThisLevel property.
Otherwise the function will return an empty string.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
	Dim obj1 As UserAndGroupCmdTarget
	Dim obj2 As UserCmdTarget
	Set obj1 = GetUserAndGroup
	If Not obj1 Is Nothing Then
		Set obj2 = obj1.GetUserObject("Group1","Ut1")
		Set obj1 = Nothing
		If Not obj2 Is Nothing Then
			MsgBox "GetXMLSettings Is -> " & obj2.GetXMLSettings,
			vbOkOnly, GetProjectTitle
			Set obj2 = Nothing
		End If
	End If
End Sub
```

IsRemote, UserCmdTarget Function

Syntax IsRemote()

Description This function returns the True boolean value when the user has executed a remote logon.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
	Dim obj1 As UserAndGroupCmdTarget
	Dim obj2 As UserCmdTarget
	Set obj1 = GetUserAndGroup
	If Not obj1 Is Nothing Then
		Set obj2 = obj1.GetUserObject("Group1","Ut1")
		Set obj1 = Nothing
		If Not obj2 Is Nothing Then
			Debug.Print obj2.IsRemote
```

```
                Set obj2 = Nothing
            End If
        End If
End Sub
```

ResetListAccessVariables, UserCmdTarget Function

Syntax ResetListAccessVariables()

Description This function refreshes the list of the variables which can be accessed according to the access level of the logged on user. This function returns a True boolean value when in error.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
        Dim obj1 As UserAndGroupCmdTarget
        Dim obj2 As UserCmdTarget
        Set obj1 = GetUserAndGroup
        If Not obj1 Is Nothing Then
            Set obj2 = obj1.GetActiveUserObject
            Set obj1 = Nothing
            If Not obj2 Is Nothing Then
                obj2.ResetListAccessVariables
                Debug.Print obj2.GetListAccessVariable
                Set obj2 = Nothing
            End If
        End If
End Sub
```

SetProp, UserCmdTarget Function

Syntax SetProp(_lpszPropID, _lpszValue)

Description This function permits you to set the value of a property for the user selected. In this way the properties maintained for all the duration of the session being operated in by the user, eg. from a web Client, can be managed. These properties are not retentive and therefore are lost when user logs off.

Parameter	Description
lpszPropID as string	ID name of property to be set
lpszValue as string	Value to be set in property

Result Boolean

Example 1:

```
'Code executable also on WebClient
Option Explicit
Public Sub Click()
	Dim objUser As UserCmdTarget

	Set objUser = GetSynopticObject.GetActiveUserObject
	If Not objUser Is Nothing Then
		objUser.SetProp("Prop1","Name")
		objUser.SetProp("Prop2","Description")
		Debug.Print "Prop1 = " & objUser.GetProp("Prop1") & "  -  " & "Prop2 =
		" & objUser.GetProp("Prop2")
		Set objUser = Nothing
	End If
End Sub
```

Example 2:

```
'Code not executable on WebClient
Option Explicit
Public Sub Click()
	Dim objUserAndGroup As UserAndGroupCmdTarget
	Dim objUser As UserCmdTarget

	Set objUserAndGroup = GetUserAndGroup
	If Not objUserAndGroup Is Nothing Then
		Set objUser = objUserAndGroup.GetActiveUserObject
		If Not objUser Is Nothing Then
			objUser.SetProp("Prop1","Name")
			objUser.SetProp("Prop2","Description")
			Debug.Print "Prop1 = " & objUser.GetProp("Prop1") & "  -  " &
			"Prop2 = " & objUser.GetProp("Prop2")
			Set objUser = Nothing
		End If
		Set objUserAndGroup = Nothing
	End If
End Sub
```

AccessLevel, UserCmdTarget Property

Syntax AccessLevel = _Long

Description This property allows you to set the User Access Level bits to be associated to the user.

This property can be accessed only when a user has logged on (by using the LogonUser' function or user has logged on from a console) with a level equal to or higher than administrator and when the reference user has a User Level lower than the value set in the AllowRuntimeChangesForUsersBelowThisLevel property. Otherwise the returned value will be equal to 1.

Parameter	Description
None	None

Result Long

Example:

```
Option Explicit
Public Sub Click()
        Dim obj1 As UserAndGroupCmdTarget
        Dim obj2 As UserCmdTarget
        Set obj1 = GetUserAndGroup
        If Not obj1 Is Nothing Then
                Set obj2 = obj1.GetUserObject("Group1","Ut1")
                Set obj1 = Nothing
                If Not obj2 Is Nothing Then
                        Debug.Print obj2.AccessLevel
                        Set obj2 = Nothing
                End If
        End If
End Sub
```

AccountDisabled, UserCmdTarget Property

Syntax AccountDisabled = _Boolean

Description This property enables or disables the User.
This property can only be accessed in write when a user has logged on (by using the 'LogonUser' function for example, or the by logging on from a console) with a level equal to or higher than administrator and when the reference user has a User Level lower than the value set in the AllowRuntimeChangesForUsersBelowThisLevel property.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
        Dim obj1 As UserAndGroupCmdTarget
        Dim obj2 As UserCmdTarget
        Set obj1 = GetUserAndGroup
        If Not obj1 Is Nothing Then
                Set obj2 = obj1.GetActiveUserObject
                Set obj1 = Nothing
                If Not obj2 Is Nothing Then
                        Debug.Print obj2.AccountDisabled
                        Set obj2 = Nothing
                End If
        End If
End Sub
```

CannotChangePassword, UserCmdTarget Property

Syntax CannotChangePassword= _Boolean

Description This property allows the option, which impedes passwords to be changed at runtime during the login phase, to be read or set at user level
To set this property from Basic Script, a user with a level equal higher than administrator level must be logged in at runtime using, for example, the 'LogonUser' function or logging in from consol.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
Dim objUserAndGroup     As UserAndGroupCmdTarget
Dim objUserGroup        As UserGroupCmdTarget
Dim objUser             As UserCmdTarget
        Set objUserAndGroup = GetUserAndGroup
        If Not objUserAndGroup Is Nothing Then
                objUserAndGroup.LogonUser("Admin","Admin")
                Set objUserGroup = objUserAndGroup.GetGroupObject("Users")
                If Not objUserGroup Is Nothing Then
                        Set objUser = objUserGroup.GetUserObject("User01")
                        objUser.CannotChangePassword = Not
                        objUser.CannotChangePassword
                End If
                LogoffActiveUser
        End If
End Sub
```

CommandListLogoff, UserCmdTarget Property

Syntax CommandListLogoff

Description This property returns a string containing the XML code of the LogOff Command List associated to the referenced user.

Parameter	Description
None	None

Result String

Example:

```
Sub Main()
        Dim objUserAndGroup As UserAndGroupCmdTarget
        Dim objUser As UserCmdTarget
        Set objUserAndGroup = GetUserAndGroup
        If Not objUserAndGroup Is Nothing Then
                objUserAndGroup.LogonUser("Administrator", "Administrator")
                Set objUser = objUserAndGroup.GetUserObject("Users","User01")
                If Not objUser Is Nothing Then
                        MsgBox "CommandListLogoff" & objUser.CommandListLogoff,
                        vbInformation, GetProjectTitle
                        Set objUser = Nothing
                End If
                Set objUserAndGroup = Nothing
        End If
End Sub
```

CommandListLogon, UserCmdTarget Property

Syntax CommandListLogon

Description This property returns a string containing the XML code of the LogOn Command List associated to the referenced user.

Parameter	Description
None	None

Result String

Example:

```
Sub Main()
        Dim objUserAndGroup As UserAndGroupCmdTarget
        Dim objUser As UserCmdTarget
        Set objUserAndGroup = GetUserAndGroup
        If Not objUserAndGroup Is Nothing Then
                objUserAndGroup.LogonUser("Administrator", "Administrator")
                Set objUser = objUserAndGroup.GetUserObject("Users","User01")
                If Not objUser Is Nothing Then
                        MsgBox "CommandListLogon = " & objUser.CommandListLogon,
                        vbInformation, GetProjectTitle
                        Set objUser = Nothing
                End If
                Set objUserAndGroup = Nothing
        End If
End Sub
```

Description, UserCmdTarget Property

Syntax Description = _String

Description This property sets or returns a descriptive text relating to the user's profile. The description of the user will be used by the system for identifying the active User, and therefore record the user wherever they are required to enter their "Electronic signature".
The property can be accessed in write only when a user has logged on (by using the 'LogonUser' function or by logging on from a console) with a level equal to or higher than administrator level and when the reference user has a User Lever lower than the value set in the AllowRuntimeChangesForUsersBelowThisLevel property.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim obj1 As UserAndGroupCmdTarget
        Dim obj2 As UserCmdTarget
        Set obj1 = GetUserAndGroup
        If Not obj1 Is Nothing Then
                Set obj2 = obj1.GetActiveUserObject
                Set obj1 = Nothing
                If Not obj2 Is Nothing Then
                        Debug.Print obj2.Description
                        Set obj2 = Nothing
                End If
        End If
End Sub
```

Email, UserCmdTarget Property

Syntax Description = _String

Description This property sets or returns the User's E-mail address for sending e-mails.
The property can be accessed in read and write only when a user has logged on (by using the 'LogonUser' function or by logging on from the console) with a lever the same as of higher than administrator level and when the reference user has a Level user lower than the value set in the AllowRuntimeChangesForUsersBelowThisLevel property.
When there are no users logged on, this property will return an empty string.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim obj1 As UserAndGroupCmdTarget
        Dim obj2 As UserCmdTarget
```

```
        Set obj1 = GetUserAndGroup
        If Not obj1 Is Nothing Then
                Set obj2 = obj1.GetUserObject("Group1","Ut1")
                Set obj1 = Nothing
                If Not obj2 Is Nothing Then
                        Debug.Print obj2.Email
                        Set obj2 = Nothing
                End If
        End If
End Sub
```

EnableAutoLoggoff, UserCmdTarget Property

Syntax EnableAutoLoggoff = _Boolean

Description This property sets or returns the AutoLogoff configuration for the reference User. When set with a True boolean value, the Logoff will be automatically executed by Movicon after the time set in the "Autologoff timeout (sec)" property. the time count will begin the moment in which user no longer continues operating the keyboard or mouse.

When the "Enable Auto Logoff" is disabled, the logged on user will remain active until deactivated (this command is setup by the programmer in the project) or until the user is replaced by another user.

This property can only be accessed in write when a user is logged on (by using the 'LongonUser' function or by logging on from a console) with a level the same as or higher than the administrator level and when the reference User Level has a higher value than the one set in the AllowRuntimeChangesForUsersBelowThisLevel property.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
        Dim obj1 As UserAndGroupCmdTarget
        Dim obj2 As UserCmdTarget
        Set obj1 = GetUserAndGroup
        If Not obj1 Is Nothing Then
                Set obj2 = obj1.GetActiveUserObject
                Set obj1 = Nothing
                If Not obj2 Is Nothing Then
                        Debug.Print obj2.EnableAutoLoggoff
                        Set obj2 = Nothing
                End If
        End If
End Sub
```

ExpiringDaysPassword, UserCmdTarget Property

Syntax ExpiringDaysPassword = _Long

Description This property allows you to set the number of days after which the User's Password will expire and will no longer be usable. Once expired, the next authentication will ask the user to enter a new password. Values entered in this field only have meaning when the "Must Change Password" user's property has been enabled. When value in this property is left set at zero, it will inherit the value set in the "Expiring Password (Days)" property from the Group it belongs to.
The property can be accessed (read/write) only when a user log on has been effected (by either using the 'LogonUser' function or the user log on from console) with a level equal to or higher than the administrator level and when the reference user has a User Lever lower than the value set in the AllowRuntimeChangesForUsersBelowThisLevel property.
Otherwise the returned value will be '0'.

Parameter	Description
None	None

Result Long

Example:

```
Option Explicit
Public Sub Click()
        Dim obj1 As UserAndGroupCmdTarget
        Dim obj2 As UserCmdTarget
        Set obj1 = GetUserAndGroup
        If Not obj1 Is Nothing Then
                Set obj2 = obj1.GetUserObject("Group1","Ut1")
                Set obj1 = Nothing
                If Not obj2 Is Nothing Then
                        Debug.Print obj2.ExpiringDaysPassword
                        Set obj2 = Nothing
                End If
        End If
End Sub
```

FaxAreaCode, UserCmdTarget Property

Syntax FaxAreaCode = _String

Description This property sets or returns the Area code relating to the user's Fax number.
The property can be accessed in read and write only when a user has logged on (by using the ' LogonUser' function or by logging on from the consul) with a level the same as or higher than administrator level and when the reference user has a User Level lower than the value set in AllowRuntimeChangesForUsersBelowThisLevel property.
Otherwise this property will return an empty string.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim obj1 As UserAndGroupCmdTarget
        Dim obj2 As UserCmdTarget
        Set obj1 = GetUserAndGroup
        If Not obj1 Is Nothing Then
                Set obj2 = obj1.GetUserObject("Group1","Ut1")
                Set obj1 = Nothing
                If Not obj2 Is Nothing Then
                        Debug.Print obj2.FaxAreaCode
                        Set obj2 = Nothing
                End If
        End If
End Sub
```

FaxCountryCode, UserCmdTarget Property

Syntax FaxCountryCode = _String

Description This property sets or returns the Country code relating to the user's Fax number. The property can be accessed in read and write only when a user has logged on (by using the ' LogonUser' function or by logging on from the consul) with a level the same as or higher than administrator level and when the reference user has a User Level lower than the value set in AllowRuntimeChangesForUsersBelowThisLevel property. Otherwise this property will return an empty string.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim obj1 As UserAndGroupCmdTarget
        Dim obj2 As UserCmdTarget
        Set obj1 = GetUserAndGroup
        If Not obj1 Is Nothing Then
                Set obj2 = obj1.GetUserObject("Group1","Ut1")
                Set obj1 = Nothing
                If Not obj2 Is Nothing Then
                        Debug.Print obj2.FaxCountryCode
                        Set obj2 = Nothing
                End If
        End If
End Sub
```

FaxPhoneNumber, UserCmdTarget Property

Syntax FaxPhoneNumber = _String

Description This property sets or returns the user's Fax phone number.
The property can be accessed in read and write only when a user has logged on (by using the ' LogonUser' function or by logging on from the consul) with a level the same as or higher than administrator level and when the reference user has a User Level lower than the value set in AllowRuntimeChangesForUsersBelowThisLevel property.
Otherwise this property will return an empty string.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim obj1 As UserAndGroupCmdTarget
        Dim obj2 As UserCmdTarget
        Set obj1 = GetUserAndGroup
        If Not obj1 Is Nothing Then
            Set obj2 = obj1.GetUserObject("Group1","Ut1")
            Set obj1 = Nothing
            If Not obj2 Is Nothing Then
                Debug.Print obj2.FaxPhoneNumber
                Set obj2 = Nothing
            End If
        End If
End Sub
```

Language, UserCmdTarget Property

Syntax Language = _String

Description This property is used for reading or setting the User's default language.
The language inserted in this property will automatically activate when user logs on. The will render the system multilanguage where each user can be set with a desired default language.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim myObject As UserAndGroupCmdTarget
        Dim myUser As UserCmdTarget
```

```
        Set myObject = GetUserAndGroup
        If Not myObject Is Nothing Then
                Set myUser = GetUserObject("Group1","User1")
                If Not myUser Is Nothing Then
                        MsgBox("User's Language = " & myUser.Language, vbOkOnly,
                        GetProjectTitle)
                End If
        End If
End Sub
```

Level, UserCmdTarget Property

Syntax Level = _Integer

Description This property sets or returns the User Level assigned to the user.

I values are:

enum_UserLevel0 = level 0
enum_UserLevel1 = level 1
enum_UserLevel2 = level 2
enum_UserLevel3 = level 3
enum_UserLevel4 = level 4
enum_UserLevel5 = level 5
enum_UserLevel6 = level 6
enum_UserLevel7 = level 7
enum_UserLevel8 = level 8
enum_UserLevelAdministrator = Administrator level
enum_UserLevelDeveloper = Developer level

This property can only be accessed when a user has logged on (either by using the 'LogonUser' or by logging on from a console) with a level equal to or higher than administrator and whether the reference user has a User Level lower than the value set in AllowRuntimeChangesForUsersBelowThisLevel property.
Otherwise the returned value will be equal to -1.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
        Dim obj1 As UserAndGroupCmdTarget
        Dim obj2 As UserCmdTarget
        Set obj1 = GetUserAndGroup
        If Not obj1 Is Nothing Then
                Set obj2 = obj1.GetUserObject("Group1","Ut1")
                Set obj1 = Nothing
                If Not obj2 Is Nothing Then
                        Debug.Print obj2.Level
                        Set obj2 = Nothing
                End If
        End If
End Sub
```

Locked, UserCmdTarget Property

Syntax Locked = _Boolean

Description This property allows you to lock out a specified user.
This property can only be accessed when a user has logged on (either by using the 'LogonUser' or by logging on from a console) with a level equal to or higher than administrator and whether the reference user has a User Level lower than the value set in AllowRuntimeChangesForUsersBelowThisLevel.
Otherwise the returned value will be equal to the False boolean value.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
        Dim obj1 As UserAndGroupCmdTarget
        Dim obj2 As UserCmdTarget
        Set obj1 = GetUserAndGroup
        If Not obj1 Is Nothing Then
                Set obj2 = obj1.GetUserObject("Group1","Ut1")
                Set obj1 = Nothing
                If Not obj2 Is Nothing Then
                        Debug.Print obj2.Locked
                        Set obj2 = Nothing
                End If
        End If
End Sub
```

LogoffScript, UserCmdTarget Property

Syntax LogoffScript = _String

Description This property sets or returns the name of the script to be run upon user LogOff.
This property can only be accessed (read/write) when a user has logged on (either by using the 'LogonUser' or by logging on from a console) with a level equal to or higher than administrator and whether the reference user has a User Level lower than the value set in AllowRuntimeChangesForUsersBelowThisLevel property.
Otherwise this property will return an empty string.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim myObject As UserAndGroupCmdTarget
```

```
        Dim myUser As UserCmdTarget
        Set myObject = GetUserAndGroup
        If Not myObject Is Nothing Then
                Set myUser = GetUserObject("Group1","Ut1")
                If Not myUser Is Nothing Then
                        MsgBox("LogoffScript is: " & myUser.LogoffScript, vbOkOnly,
                        GetProjectTitle)
                End If
        End If
End Sub
```

LogonScript, UserCmdTarget Property

Syntax LogonScript = _String

Description This property sets or returns the name of the script to be run up User LogOn.
This property can only be accessed (read write) when a user has logged on (either by using the 'LogonUser' or by logging on from a console) with a level equal to or higher than administrator and whether the reference user has a User Level lower than the value set in AllowRuntimeChangesForUsersBelowThisLevel property.
Otherwise this property returns an empty string.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim myObject As UserAndGroupCmdTarget
        Dim myUser As UserCmdTarget
        Set myObject = GetUserAndGroup
        If Not myObject Is Nothing Then
                Set myUser = GetUserObject("Group1","Ut1")
                If Not myUser Is Nothing Then
                        MsgBox("LogonScript is: " & myUser.LogonScript, vbOkOnly,
                        GetProjectTitle)
                End If
        End If
End Sub
```

MobileAreaCode, UserCmdTarget Property

Syntax MobileAreaCode = _String

Description This property sets or returns the Area Code relating to the User's mobile phone number.
This property can be accessed (read/write) only when a user has logged on (using the 'LogonUser' function or user logon from consol) with a level equal to or higher than administrator level and when the reference user has a User level higher to the value set in the AllowRuntimeChangesForUsersBelowThisLevel property.
If otherwise the property will return an empty string.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
	Dim myObject As UserAndGroupCmdTarget
	Dim myUser As UserCmdTarget
	Set myObject = GetUserAndGroup
	If Not myObject Is Nothing Then
		Set myUser = GetUserObject("Group1","Ut1")
		If Not myUser Is Nothing Then
			MsgBox("MobileAreaCode is: " & myUser.MobileAreaCode, vbOkOnly,
			GetProjectTitle)
		End If
	End If
End Sub
```

MobileCountryCode, UserCmdTarget Property

Syntax MobileCountryCode = _String

Description This property sets or returns the Country Code relating to the User's mobile phone number.
This property can be accessed (read/write) only when a user has logged on (using the 'LogonUser' function or user logon from consol) with a level equal to or higher than administrator level and when the reference user has a User level higher to the value set in the AllowRuntimeChangesForUsersBelowThisLevel property.
If otherwise the property will return an empty string.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
	Dim myObject As UserAndGroupCmdTarget
	Dim myUser As UserCmdTarget
	Set myObject = GetUserAndGroup
	If Not myObject Is Nothing Then
		Set myUser = GetUserObject("Group1","Ut1")
		If Not myUser Is Nothing Then
			MsgBox("MobileCountryCode is: " & myUser.MobileCountryCode,
			vbOkOnly, GetProjectTitle)
		End If
	End If
End Sub
```

MobilePhoneNumber, UserCmdTarget Property

Syntax MobilePhoneNumber = _String

Description This property sets or returns the User's Mobile phone number.
This property can be accessed (read/write) only when a user has logged on (using the 'LogonUser' function or user logon from consol) with a level equal to or higher than administrator level and when the reference user has a User level higher to the value set in the AllowRuntimeChangesForUsersBelowThisLevel property.
If otherwise the property will return an empty string.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim myObject As UserAndGroupCmdTarget
        Dim myUser As UserCmdTarget
        Set myObject = GetUserAndGroup
        If Not myObject Is Nothing Then
                Set myUser = GetUserObject("Group1","Ut1")
                If Not myUser Is Nothing Then
                        MsgBox("MobilePhoneNumber is: " & myUser.MobilePhoneNumber,
                        vbOkOnly, GetProjectTitle)
                End If
        End If
End Sub
```

MustChangedPasswordLogon, UserCmdTarget Property

Syntax MustChangedPasswordLogon = _Boolean

Description This property allows a specific user's access to be blocked.
This property can only be accessed when a user has logged on (either by using the 'LogonUser' or by logging on from a console) with a level equal to or higher than administrator and whether the reference user has a User Level lower than the value set in AllowRuntimeChangesForUsersBelowThisLevel property.
Otherwise the returned value will be equal to the False boolean value.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
        Dim obj1 As UserAndGroupCmdTarget
        Dim obj2 As UserCmdTarget
        Set obj1 = GetUserAndGroup
        If Not obj1 Is Nothing Then
                Set obj2 = obj1.GetUserObject("Group1","Ut1")
                Set obj1 = Nothing
                If Not obj2 Is Nothing Then
                        Debug.Print obj2.MustChangedPasswordLogon
                        Set obj2 = Nothing
                End If
        End If
End Sub
```

Name, UserCmdTarget Property

Syntax Name = _String

Description This property sets or returns the User's name.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim myObject As UserAndGroupCmdTarget
        Dim myUser As UserCmdTarget
        Set myObject = GetUserAndGroup
        If Not myObject Is Nothing Then
                Set myUser = GetUserObject("Group1","Ut1")
                If Not myUser Is Nothing Then
                        MsgBox("Name is: " & myUser.Name, vbOkOnly, GetProjectTitle)
                End If
        End If
End Sub
```

OnLine, UserCmdTarget Property

Syntax OnLine = _Boolean

Description This property returns the True boolean value when the reference User is OnLine (has logged on).
This property can be accessed (read/write) only when a user has logged on (using the 'LogonUser' function or user logon from consol) with a level equal to or higher than administrator level and when the reference user has a User level higher to the value set in the AllowRuntimeChangesForUsersBelowThisLevel property. When set with a True, the reference user will be automatically Logged on.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
        Dim obj1 As UserAndGroupCmdTarget
        Dim obj2 As UserCmdTarget
        Set obj1 = GetUserAndGroup
        If Not obj1 Is Nothing Then
                Set obj2 = obj1.GetUserObject("Group1","Ut1")
                Set obj1 = Nothing
                If Not obj2 Is Nothing Then
                        Debug.Print obj2.OnLine
                        Set obj2 = Nothing
                End If
        End If
End Sub
```

Password, UserCmdTarget Property

Syntax Password = _String

Description This property allows you to set a user password.
This property is accessible (in write) only if a user has logged in (by using the ""LogonUser" function or logging in from a console) with a level equal to or higher than administrator level (1023) and if the referenced user has a User Level equal or lower than the value set with AllowRuntimeChangesForUsersBelowThisLevel property.
When read this property always returns an empty string. The "GetXMLSettings" function can be used to view the user's properties and therefore their associated password.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
                        Dim objUserAndGroup As UserAndGroupCmdTarget
        Dim objUser As UserCmdTarget
        Dim sNewPassword As String

        sNewPassword = InputBox("Insert New Password:",GetProjectTitle,"")
        Set objUserAndGroup = GetUserAndGroup
        objUserAndGroup.LogonUser("Administrator","Administrator")
        objUserAndGroup.AllowRuntimeChangesForUsersBelowThisLevel = 8
        If Not objUserAndGroup Is Nothing Then
                Set objUser = objUserAndGroup.GetUserObject("Group1","User01")
```

```
                If Not objUser Is Nothing Then
                        objUser.Password = sNewPassword
                End If
                Set objUserAndGroup = Nothing
        End If
        LogoffActiveUser
End Sub
```

SecsAutoLoggoffTimeout, UserCmdTarget Property

Syntax SecsAutoLoggoffTimeout = _Long

Description This property sets or returns the time after which Movicon will execute the Auto Logoff of any active User. This setting only has meaning when the "Enable Auto Logoff" property has been activated. The time count starts the moment the user stops executing operations with the keyboard or mouse.

This property can only be accessed when a user has logged on (by using the 'LogonUser' function or logging on from console) with a level equal to or higher than administrator level and when the reference user has a lower User Level than the one set in the AllowRuntimeChangesForUsersBelowThisLevel property.
If this is not so, a 0 value will be returned.

Parameter	Description
None	None

Result Long

Example:

```
Option Explicit
Public Sub Click()
        Dim obj1 As UserAndGroupCmdTarget
        Dim obj2 As UserCmdTarget
        Set obj1 = GetUserAndGroup
        If Not obj1 Is Nothing Then
                Set obj2 = obj1.GetUserObject("Group1","Ut1")
                Set obj1 = Nothing
                If Not obj2 Is Nothing Then
                        Debug.Print obj2.SecsAutoLoggoffTimeout
                        Set obj2 = Nothing
                End If
        End If
End Sub
```

VoiceAreaCode, UserCmdTarget Property

Syntax VoiceAreaCode = _String

Description This property sets or returns the Area Code of the number relating to the User's voice messages.
This property can only be accessed in (read/write) when a user has logged on (with the 'LogUser' or user logon form consol) with a with a level equal to or

higher than administrator leve and when the reference user has a User Level lower than the value set in the AllowRuntimeChangesForUsersBelowThisLevel property. Otherwise this property will return an empty string.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim myObject As UserAndGroupCmdTarget
        Dim myUser As UserCmdTarget
        Set myObject = GetUserAndGroup
        If Not myObject Is Nothing Then
                Set myUser = GetUserObject("Group1","Ut1")
                If Not myUser Is Nothing Then
                        MsgBox("VoiceAreaCode is: " & myUser.VoiceAreaCode, vbOkOnly,
                        GetProjectTitle)
                End If
        End If
End Sub
```

VoiceCountryCode, UserCmdTarget Property

Syntax VoiceCountryCode = _String

Description This property sets or returns the country code relating to the User's vocal messages.
This property can only be accessed in (read/write) when a user has logged on (with the 'LogUser' or user logon form consol) with a with a level equal to or higher than administrator leve and when the reference user has a User Level lower than the value set in the AllowRuntimeChangesForUsersBelowThisLevel property. Otherwise this property will return an empty string.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim myObject As UserAndGroupCmdTarget
        Dim myUser As UserCmdTarget
        Set myObject = GetUserAndGroup
        If Not myObject Is Nothing Then
                Set myUser = GetUserObject("Group1","Ut1")
                If Not myUser Is Nothing Then
                        MsgBox("VoiceCountryCode is: " & myUser.VoiceCountryCode ,
                        vbOkOnly, GetProjectTitle)
                End If
```

```
        End If
End Sub
```

VoicePhoneNumber, UserCmdTarget Property

Syntax VoicePhoneNumber = _String

Description This property sets or returns the number for the User's voice messages. This property can only be accessed in (read/write) when a user has logged on (with the 'LogUser' or user logon form consol) with a with a level equal to or higher than administrator leve and when the reference user has a User Level lower than the value set in the AllowRuntimeChangesForUsersBelowThisLevel property. Otherwise this property will return an empty string.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim myObject As UserAndGroupCmdTarget
        Dim myUser As UserCmdTarget
        Set myObject = GetUserAndGroup
        If Not myObject Is Nothing Then
                Set myUser = GetUserObject("Group1","Ut1")
                If Not myUser Is Nothing Then
                        MsgBox("VoicePhoneNumber is: " & myUser.VoicePhoneNumber ,
                        vbOkOnly, GetProjectTitle)
                End If
        End If
End Sub
```

WebClientAutoLogoffSecs, UserCmdTarget Property

Syntax WebClientAutoLogoffSecs = _Long

Description This property allows you to read or set the Auto Log off time of the Web Client user connected to the Server application. The value set in the user's 'Web Client Autologoff (sec.)' property is taken into consideration if not null with the Password Management active. To set this property form Basic Scrip, a user with user level equal to or higher than administrator needs to logged on in runtime using the 'LogonUser' function or the the log in of a user from console.

Parameter	Description
None	None

Result Long

Example:

```
Option Explicit
Public Sub Click()
Dim objUserAndGroup  As UserAndGroupCmdTarget
Dim objUserGroup  As UserGroupCmdTarget
Dim objUser  As UserCmdTarget
        Set objUserAndGroup = GetUserAndGroup
        If Not objUserAndGroup Is Nothing Then
                objUserAndGroup.LogonUser("Admin","Admin")
                Set objUserGroup = objUserAndGroup.GetGroupObject("Users")
                If Not objUserGroup Is Nothing Then
                        Set objUser = objUserGroup.GetUserObject("User01")
                        objUser.WebClientAutoLogoffSecs = 90
                End If
                LogoffActiveUser
        End If
End Sub
```

58. UserGroupCmdTarget

GetNumUsers, UserGroupCmdTarget Function

Syntax GetNumUsers()

Description This function returns the number of user belonging to the reference group.

Note:
The function and the properties of the user group can only be accessed when a user has logged on (either by using the 'LogonUser' function or logging on from a console) with a level equal to or higher than administrator.

Parameter	Description
None	None

Result Integer

Example1:

```
Option Explicit
Sub Click()
        Dim MyGroup As UserGroupCmdTarget
        Dim MyUsersAndGroups As UserAndGroupCmdTarget
        Dim MyUser As UserCmdTarget
        Dim sUsersList As String
        Dim numUsers As Integer
        Dim i As Integer
        Dim tmpUsers As String

        'Login Administrator level
        GetPasswordFromLevel(1023)
        Set MyUsersAndGroups = GetUserAndGroup
        If Not MyUsersAndGroups Is Nothing Then
                Set MyGroup = MyUsersAndGroups.GetGroupObject("Users")
                If Not MyGroup Is Nothing Then
                        i = 0
                        numUsers = CInt(MyGroup.GetNumUsers) - 1
        For i = 0 To numUsers Step 1
                                Set MyUser = MyGroup.GetUserAtPos(i)
                                If Not MyUser Is Nothing Then
                                        sUsersList = sUsersList & MyUser.Name & "|"
                                        tmpUsers = tmpUsers & MyUser.Name & vbCrLf
                                        Set MyUser = Nothing
                                End If
                        Next i
                        MsgBox(tmpUsers,vbInformation + vbOkOnly,"Users List")
                End If
        End If
        LogoffActiveUser
        Set MyUser = Nothing
        Set MyGroup = Nothing
        Set MyUsersAndGroups = Nothing
End Sub
```

Example2:

```
Sub Click()
        Dim MyGroup As UserGroupCmdTarget
        Dim MyUsersAndGroups As UserAndGroupCmdTarget
        Dim MyUser As UserCmdTarget
        Dim sUsersList As String
        Dim numUsers As Integer
        Dim i As Integer
        Dim tmpUsers As String

        Set MyUsersAndGroups = GetUserAndGroup
        If Not MyUsersAndGroups Is Nothing Then
                'administrator login for use the group functions
                MyUsersAndGroups.LogonUser("Daniele","Daniele")
                Set MyGroup = MyUsersAndGroups.GetGroupObject("Users")
                If Not MyGroup Is Nothing Then
                        i = 0
                        numUsers = CInt(MyGroup.GetNumUsers) - 1
                        For i = 0 To numUsers Step 1
                                Set MyUser = MyGroup.GetUserAtPos(i)
                                If Not MyUser Is Nothing Then
                                        sUsersList = sUsersList & MyUser.Name & "|"
                                        tmpUsers = tmpUsers & MyUser.Name & vbCrLf
                                        Set MyUser = Nothing
                                End If
                        Next i
                        MsgBox(tmpUsers,vbInformation + vbOkOnly,"Users List")
                End If
        End If
        LogoffActiveUser
        Set MyUser = Nothing
        Set MyGroup = Nothing
        Set MyUsersAndGroups = Nothing
End Sub
```

GetUserAtPos, UserGroupCmdTarget Function

Syntax GetUserAtPos(_Position)

Description This function returns the user belonging to the group referenced at the same position passed as parameter.

Note:
The function and the properties of the user group can only be accessed when a user has logged on (either by using the 'LogonUser' function or logging on from a console) with a level equal to or higher than administrator.

Parameter	Description
Position As Integer	Reference index of user.

Result Object
If Function has been executed successfully it will retrieve an object of type UserCmdTarget if otherwise Nothing is returned.

Example:

```
Option Explicit
Public Sub Click()
        Dim MyGroup As UserGroupCmdTarget
```

```
        Dim MyUsersAndGroups As UserAndGroupCmdTarget
        Dim MyUser As UserCmdTarget
        GetPasswordFromLevel(0)
        Set MyUsersAndGroups = GetUserAndGroup
        If Not MyUsersAndGroups Is Nothing Then
                Set MyGroup = MyUsersAndGroups.GetGroupObject("Users")
                If Not MyGroup Is Nothing Then
                        Dim numUsers As Integer
                        Dim i As Integer
                        i = 0
                        numUsers = CInt(MyGroup.GetNumUsers) - 1
                        For i = 0 to numUsers Step 1
                          Set MyUser = MyGroup.GetUserAtPos(i)
                          If Not MyUser Is Nothing Then
                                  Debug.Print MyUser.Name
                                  Set MyUser = Nothing
                          End If
                        Next i
                End If
        End If

        Set MyUser = Nothing
        Set MyGroup = Nothing
        Set MyUsersAndGroups = Nothing
End Sub
```

GetUserObject, UserGroupCmdTarget Function

Syntax GetUserObject(_lpszUserName)

Description This function returns the user belonging to the referene user group. This function returns a Nothing when the reference use does not exist in the group.

Note:
The function and the properties of the user group can only be accessed when a user has logged on (either by using the 'LogonUser' function or logging on from a console) with a level equal to or higher than administrator.

Parameter	Description
lpszUserName As String	Name of user to be fetched

Result Object
If Function has been executed successfully it will retrieve an object of type UserCmdTarget if otherwise Nothing is returned.

Example:

```
Option Explicit
Public Sub Click()
        Dim MyGroup As UserGroupCmdTarget
        Dim MyUsersAndGroups As UserAndGroupCmdTarget
        Dim MyUser As UserCmdTarget
        GetPasswordFromLevel(0)
        Set MyUsersAndGroups = GetUserAndGroup
        If Not MyUsersAndGroups Is Nothing Then
                Set MyGroup = MyUsersAndGroups.GetGroupObject("Users")
                If Not MyGroup Is Nothing Then
                        Set MyUser = MyGroup.GetUserObject("Guest")
                        If Not MyUser In Nothing Then
                                MsgBox MyUser.GetXMLSettings,vbOkOnly,""
```

```
                End If
            End If
        End If
        Set MyUser = Nothing
        Set MyGroup = Nothing
        Set MyUsersAndGroups = Nothing
End Sub
```

GetXMLSettings, UserGroupCmdTarget Function

Syntax GetXMLSettings

Description This function returns a string the the contents of the project's XML file relating to the referenced group.

Note:
The function and the properties of the user group can only be accessed when a user has logged on (either by using the 'LogonUser' function or logging on from a console) with a level equal to or higher than administrator.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim MyGroup As UserGroupCmdTarget
        Dim MyUsersAndGroups As UserAndGroupCmdTarget
        Dim MyUser As UserCmdTarget
        GetPasswordFromLevel(0)
        Set MyUsersAndGroups = GetUserAndGroup
        If Not MyUsersAndGroups Is Nothing Then
            Set MyGroup = MyUsersAndGroups.GetGroupObject("Users")
            If Not MyGroup Is Nothing Then
                MsgBox MyGroup.GetXMLSettings,vbOkOnly,""
            End If
        End If
        Set MyUser = Nothing
        Set MyGroup = Nothing
        Set MyUsersAndGroups = Nothing
End Sub
```

CommandListLogoff, UserGroupCmdTarget Property

Syntax CommandListLogoff

Description This property returns a string containing the XML code of the LogOff Command List associated to the referenced group.

Parameter	Description
None	None

Result String

Example:

```
Sub Main()
        Dim objUserAndGroup As UserAndGroupCmdTarget
        Dim objGroup As UserGroupCmdTarget
        Set objUserAndGroup = GetUserAndGroup
        If Not objUserAndGroup Is Nothing Then
                objUserAndGroup.LogonUser("Administrator", "Administrator")
                Set objGroup = objUserAndGroup.GetGroupObject("Users")
                If Not objGroup Is Nothing Then
                        MsgBox "CommandListLogoff = " & objGroup.CommandListLogoff,
                        vbInformation, GetProjectTitle
                        Set objGroup = Nothing
                End If
                Set objUserAndGroup = Nothing
        End If
End Sub
```

CommandListLogon, UserGroupCmdTarget Property

Syntax CommandListLogon

Description This property returns a string containing the XML code of the LogOn Command List associated to the referenced group.

Parameter	Description
None	None

Result String

Example:

```
Sub Main()
        Dim objUserAndGroup As UserAndGroupCmdTarget
        Dim objGroup As UserGroupCmdTarget
        Set objUserAndGroup = GetUserAndGroup
        If Not objUserAndGroup Is Nothing Then
                objUserAndGroup.LogonUser("Administrator", "Administrator")
                Set objGroup = objUserAndGroup.GetGroupObject("Users")
                If Not objGroup Is Nothing Then
                        MsgBox "CommandListLogon = " & objGroup.CommandListLogon,
                        vbInformation, GetProjectTitle
                        Set objGroup = Nothing
                End If
                Set objUserAndGroup = Nothing
        End If
End Sub
```

DefaultAccessLevel, UserGroupCmdTarget Property

Syntax DefaultAccessLevel = _Long

Description This property sets or returns the Default Access Level assigned to the users belonging to the reference group.

Note:
The functions and the properties of the user groups can be accessed only when a user has logged on (by using the 'LogonUser' function or by logging on from a console) with a level equal to or higher than administrator level.

Parameter	Description
None	None

Result Long

Example:

```
Option Explicit
Public Sub Click()
        Dim obj1 As UserAndGroupCmdTarget
        Dim obj2 As UserGroupCmdTarget
        Set obj1 = GetUserAndGroup
        If Not obj1 Is Nothing Then
                Set obj2 = obj1.GetGroupObject("Group1")
                Set obj1 = Nothing
                If Not obj2 Is Nothing Then
                        Debug.Print obj2.DefaultAccessLevel
                        Set obj2 = Nothing
                End If
        End If
End Sub
```

DefaultEnableAutoLoggoff, UserGroupCmdTarget Property

Syntax DefaultEnableAutoLoggoff = _Boolean

Description This property sets or returns the AutoLogoff configuration for Users belonging to the reference Group. When set to a True boolean value, the Logoff will be executed automatically by Movicon after the time set in the "Auto logoff Timeout (sec)" property. The time count will begin the moment in which the user no longer carries out any operations from the keyboard or with the mouse.

Note:
The functions and the properties of the user groups can be accessed only when a user has logged on (by using the 'LogonUser' function or by logging on from a console) with a level equal to or higher than administrator level.

Parameter	Description
None	None

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
	Dim obj1 As UserAndGroupCmdTarget
	Dim obj2 As UserGroupCmdTarget
	Set obj1 = GetUserAndGroup
	If Not obj1 Is Nothing Then
		Set obj2 = obj1.GetGroupObject("Group1")
		Set obj1 = Nothing
		If Not obj2 Is Nothing Then
			Debug.Print obj2.DefaultEnableAutoLoggoff
			Set obj2 = Nothing
		End If
	End If
End Sub
```

DefaultExpiringDaysPassword, UserGroupCmdTarget Property

Syntax DefaultExpiringDaysPassword = _Long

Description This property allows the number of default days to be set after which the Passwords of the Users belonging to the reference Group expires and will not longer be useable. Once expired, the next user authentication will be requested to insert a new password. The value set in this property only has meaning when the "Must Change Password" User property has been enabled. When the value of this value is left at zero, the value set in the "Expiring Password (Days)" property, of the group which the user belongs to, will be used.

Note:
The functions and the properties of the user groups can be accessed only when a user has logged on (by using the 'LogonUser' function or by logging on from a console) with a level equal to or higher than administrator level.

Parameter	Description
None	None

Result Long

Example:

```
Option Explicit
Public Sub Click()
	Dim obj1 As UserAndGroupCmdTarget
	Dim obj2 As UserGroupCmdTarget
	Set obj1 = GetUserAndGroup
	If Not obj1 Is Nothing Then
		Set obj2 = obj1.GetGroupObject("Group1")
		Set obj1 = Nothing
		If Not obj2 Is Nothing Then
			Debug.Print obj2.DefaultExpiringDaysPassword
			Set obj2 = Nothing
		End If
	End If
End Sub
```

DefaultLevel, UserGroupCmdTarget Property

Syntax DefaultLevel = _Integer

Description This property sets or resets the Default Access Level to be associated to the Group.

The possible values for these levels are:

enum_UserLevel0 = level 0
enum_UserLevel1 = level 1
enum_UserLevel2 = level 2
enum_UserLevel3 = level 3
enum_UserLevel4 = level 4
enum_UserLevel5 = level 5
enum_UserLevel6 = level 6
enum_UserLevel7 = level 7
enum_UserLevel8 = level 8
enum_UserLevelAdministrator = administrator level
enum_UserLevelDeveloper = Developer level

Note:
The functions and the properties of the user groups can be accessed only when a user has logged on (by using the 'LogonUser' function or by logging on from a console) with a level equal to or higher than administrator level.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
        Dim obj1 As UserAndGroupCmdTarget
        Dim obj2 As UserGroupCmdTarget
        Set obj1 = GetUserAndGroup
        If Not obj1 Is Nothing Then
                Set obj2 = obj1.GetGroupObject("Group1")
                Set obj1 = Nothing
                If Not obj2 Is Nothing Then
                        Debug.Print obj2.DefaultLevel
                        Set obj2 = Nothing
                End If
        End If
End Sub
```

DefaultLogoffScript, UserGroupCmdTarget Property

Syntax DefaultLogoffScript = _String

Description This property sets or returns the name of the script to be executed when Users belonging to the reference Group log off.

Note:
The functions and the properties of the user groups can be accessed only when a user has logged on (by using the 'LogonUser' function or by logging on from a console) with a level equal to or higher than administrator level.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim obj1 As UserAndGroupCmdTarget
        Dim obj2 As UserGroupCmdTarget
        Set obj1 = GetUserAndGroup
        If Not obj1 Is Nothing Then
                Set obj2 = obj1.GetGroupObject("Group1")
                Set obj1 = Nothing
                If Not obj2 Is Nothing Then
                        Debug.Print obj2.DefaultLogoffScript
                        Set obj2 = Nothing
                End If
        End If
End Sub
```

DefaultLogonScript, UserGroupCmdTarget Property

Syntax DefaultLogonScript = _String

Description This property sets or returns the name of the script to executed when the Users belonging to the reference group log off.

Note:
The functions and the properties of the user groups can be accessed only when a user has logged on (by using the 'LogonUser' function or by logging on from a console) with a level equal to or higher than administrator level.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim obj1 As UserAndGroupCmdTarget
        Dim obj2 As UserGroupCmdTarget
        Set obj1 = GetUserAndGroup
```

```
        If Not obj1 Is Nothing Then
                Set obj2 = obj1.GetGroupObject("Group1")
                Set obj1 = Nothing
                If Not obj2 Is Nothing Then
                        Debug.Print obj2.DefaultLogonScript
                        Set obj2 = Nothing
                End If
        End If
End Sub
```

DefaultSecsAutoLoggoffTimeout, UserGroupCmdTarget Property

Syntax DefaultSecsAutoLoggoffTimeout = _Integer

Description This property sets or returns the time after which Movicon will execute the Auto Logoff of any active User belonging to the reference group. This setting only has meaning when the "Enable Auto Logoff" property has been activated. The time count will begin the moment in which the user no longer carries out any operations either from the keyboard or with the mouse.

Note:
The functions and the properties of the user groups can be accessed only when a user has logged on (by using the 'LogonUser' function or by logging on from a console) with a level equal to or higher than administrator level.

Parameter	Description
None	None

Result Integer

Example:

```
Option Explicit
Public Sub Click()
        Dim obj1 As UserAndGroupCmdTarget
        Dim obj2 As UserGroupCmdTarget
        Set obj1 = GetUserAndGroup
        If Not obj1 Is Nothing Then
                Set obj2 = obj1.GetGroupObject("Group1")
                Set obj1 = Nothing
                If Not obj2 Is Nothing Then
                        Debug.Print obj2.DefaultSecsAutoLoggoffTimeout
                        Set obj2 = Nothing
                End If
        End If
End Sub
```

Description, UserGroupCmdTarget Property

Syntax Description = _String

Description This property sets or returns a descriptive text related to the reference Group. The description is used only as a reminder for the programmer and appears only in the Group's "General Properties".

Note:
The functions and the properties of the user groups can be accessed only when a user has logged on (by using the 'LogonUser' function or by logging on from a console) with a level equal to or higher than administrator level.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim obj1 As UserAndGroupCmdTarget
        Dim obj2 As UserGroupCmdTarget
        Set obj1 = GetUserAndGroup
        If Not obj1 Is Nothing Then
            Set obj2 = obj1.GetGroupObject("Group1")
            Set obj1 = Nothing
            If Not obj2 Is Nothing Then
                Debug.Print obj2.Description
                Set obj2 = Nothing
            End If
        End If
End Sub
```

Language, UserGroupCmdTarget Property

Syntax Language = _String

Description Questa proprietà permette di leggere o impostare la lingua di default per il Gruppo Utenti selezionato. Impostando la lingua a livello di Gruppo la "Lingua" di default sarà valida per tutti gli Utenti appartenenti al Gruppo e che non hanno la stessa proprietà impostata.
Inserendo una lingua in questa proprietà quando l'Utente appartenente al Gruppo esegue il Logon verrà automaticamente attivata quella lingua, a meno che la stessa proprietà "Language" non sia stata definita anche a livello dell'Utente, nel qual caso avrà la priorità la proprietà dell'Utente.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim obj1 As UserAndGroupCmdTarget
        Dim oGroup1 As UserGroupCmdTarget

        Set obj1 = GetUserAndGroup
        If Not obj1 Is Nothing Then
```

```
                Set oGroup1 = obj1.GetGroupObject("Group1")
                Set obj1 = Nothing
                If Not oGroup1 Is Nothing Then
                        MsgBox("Group Language = " & oGroup1.Language, vbOkOnly,
                        GetProjectTitle)
                        Set obj2 = Nothing
                End If
        End If
End Sub
```

Name, UserGroupCmdTarget Property

Syntax Name = _String

Description This property sets or returns the name of the reference Group.

Note that:
The functions and properties of the User groups can only be accessed when a user has logged on (either with the 'LogonUser' function or logging on from consol) with a level equal or higher than administrator level.

Parameter	Description
None	None

Result String

Example:

```
Option Explicit
Public Sub Click()
        Dim obj1 As UserAndGroupCmdTarget
        Dim obj2 As UserGroupCmdTarget
        Set obj1 = GetUserAndGroup
        If Not obj1 Is Nothing Then
                Set obj2 = obj1.GetGroupObject("Group1")
                Set obj1 = Nothing
                If Not obj2 Is Nothing Then
                        Debug.Print obj2.Name
                        Set obj2 = Nothing
                End If
        End If
End Sub
```

WebClientAutoLogoffSecs, UserGroupCmdTarget Property

Syntax WebClientAutoLogoffSecs = _Long

Description This property allows you to read or set the Auto Log off time of the Web Client user connected to the Server application. The value set in the use group is taken into consideration if the user's 'Web Client Autologoff (sec.)' property is null with the Password Management active. To set this property form Basic Scrip, a user with user level equal to or higher than administrator needs to logged on in runtime using the 'LogonUser' function or the the log in of a user from console.

Parameter	Description
None	None

Result Long

Example:

```
Option Explicit
Public Sub Click()
Dim objUserAndGroup As UserAndGroupCmdTarget
Dim objUserGroup As UserGroupCmdTarget
        Set objUserAndGroup = GetUserAndGroup
        If Not objUserAndGroup Is Nothing Then
                objUserAndGroup.LogonUser("Admin","Admin")
                Set objUserGroup = objUserAndGroup.GetGroupObject("Users")
                objUserGroup.WebClientAutoLogoffSecs = 90
                LogoffActiveUser
        End If
 End Sub
```

59. WorkspaceCmdTarget

OpenProject, WorkspaceCmdTarget Function

Syntax OpenProject(_lpszFilePath)

Description This function opens the project relating to the path passed as parameter. This function can only be executed in design mode.

Parameter	Description
lpszFilePath As String	Project path to be opened.

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
        OpenProject("C:\Movicon\Mov1.movprj")
End Sub
```

OpenScreen, WorkspaceCmdTarget Function

Syntax OpenScreen(_lpszScreenName)

Description This function opens the referene screen. This function can only be executed in design mode.

Parameter	Description
lpszScreenName As String	Screen resource name

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
        OpenScript("Screen1")
End Sub
```

OpenScript, WorkspaceCmdTarget Function

Syntax OpenScript(_lpszScriptName)

Description This function opens the reference script. This function can only be executed in design mode.

Parameter	Description
lpszScriptName As String	Script resource name

Result Boolean

Example:

```
Option Explicit
Public Sub Click()
        OpenScript("Script1")
End Sub
```

MOVICON 11

Movicon™ is a trademark of Progea, related to the HMI/SCADA platform entirely developed and produced by Progea. © 2012 All Rights reserved.
No part of this document or of the program may be reproduced or transmitted in any form without the express written permission of Progea.
Information in this document is subject to change without notice and is not binding in any way for the company producing it.

Via S.Anna, 88/E
41100 Modena - Italy
Tel. +39 059 451060
Fax +39 059 451061
Email:info@progea.com
Http://www.progea.com

Via XX Settembre, 30
Tecnocity Alto Milanese
20025 Legnano (MI) Italy
Tel. +39 0331 486653
Fax +39 0331 455179
Email: willems@progea.com

Progea Deutschland GmbH
Marie-Curie-Str. 12
D-78048 VS-Villingen
Tel: +49 (0) 7721 / 99 25 992
Fax: +49 (0) 7721 / 99 25 993
info@progea.de

Progea International Ltd
via Penate 16
6850 Mendrisio - Switzerland
tel +41 (91) 9676610
fax +41 (91) 9676611
international@progea.com

Progea USA LLC
2800 East Enterprise Avenue
Appleton, WI 54914
Tel. +1 (888) 305 2999
Fax. +1 (920) 257 4213
info@progea.us