

SQABasic Language Reference

Version 2000.02.10

SQABasic Language Reference

Copyright © 1998-2000 Rational Software Corporation. All rights reserved. The contents of this manual and the associated software are the property of Rational Software Corporation and are copyrighted. Any reproduction in whole or in part is strictly prohibited. For additional copies of this manual or software, please contact Rational Software Corporation.

Rational, the Rational logo, PerformanceStudio, SiteCheck, TestFactory, TestStudio, Object-Oriented Recording, and Object Testing are trademarks or registered trademarks of Rational Software Corporation in the United States and in other countries. Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. All other names are used for identification purposes only and are trademarks or registered trademarks of their respective companies.

U.S. GOVERNMENT RIGHTS. Use, duplication, or disclosure by the U.S. Government is subject to restrictions set forth in the applicable Rational License Agreement and in DFARS 227.7202-1(a) and 227.7202-3(a) (1995), DFARS 252.227-7013(c)(1)(ii) (Oct 1988), FAR 12.212(a) 1995, FAR 52.227-19, or FAR 52.227-14, as applicable.

Revised 4/2000

This manual prepared by:
Rational Software Corporation
20 Maguire Road
Lexington, MA 02421
U.S.A.

Phone:
800-433-5444
408-863-4000

E-mail: support@rational.com
Web: <http://www.rational.com>

P/N 800-023367-000

▶ ▶ ▶ Contents

Preface

| | |
|---|------|
| Audience..... | xix |
| Other Resources..... | xix |
| Accessing SQABasic Help..... | xix |
| Typographical Conventions..... | xxi |
| Contacting Rational Technical Publications..... | xxii |
| Contacting Rational Technical Support | xxii |

Part I Introducing SQABasic

| | |
|---|-----|
| 1 What Is SQABasic? | |
| Automatic Script Generation | 1-1 |
| Working with Scripts..... | 1-2 |
| Your Work Environment | 1-2 |
| Source and Runtime Files..... | 1-3 |
| SQABasic Additions to the Basic Language..... | 1-3 |
| Other Commands Not Found in Basic..... | 1-4 |
| VU Scripting Language | 1-4 |
| 2 Functional List | |
| Arrays..... | 2-1 |
| Compiler Directives..... | 2-1 |
| Datapool Commands (SQABasic Additions) | 2-2 |
| Dates & Times..... | 2-2 |
| Declarations..... | 2-2 |
| Dialog Box Definition..... | 2-3 |

Contents

| | |
|--|------|
| Dialog Box Services..... | 2-4 |
| Disk and Directory Control..... | 2-4 |
| Dynamic Data Exchange (DDE) | 2-5 |
| Environmental Control..... | 2-5 |
| Error Handling..... | 2-5 |
| File Control..... | 2-6 |
| File Input/Output..... | 2-6 |
| Financial Functions | 2-7 |
| Flow Control..... | 2-7 |
| Numeric and Trigonometric Functions..... | 2-8 |
| Object Scripting Commands (SQABasic Additions) | 2-8 |
| Objects..... | 2-9 |
| ODBC | 2-9 |
| Screen Input/Output..... | 2-10 |
| SQABasic Commands | 2-10 |
| String Conversions..... | 2-10 |
| String Manipulation | 2-11 |
| Timing and Coordination Commands (SQABasic Additions)..... | 2-12 |
| User Action Commands (SQABasic Additions) | 2-12 |
| Utility Commands (SQABasic Additions) | 2-14 |
| Variants..... | 2-17 |
| Verification Point Commands (SQABasic Additions) | 2-17 |

Part II Using SQABasic

3 SQABasic Fundamentals

| | |
|---|-----|
| Commands..... | 3-2 |
| Arguments..... | 3-3 |
| Passing Arguments By Value or By Reference..... | 3-3 |
| Passing Named Arguments..... | 3-4 |
| Data Types..... | 3-5 |
| Descriptions of SQABasic Data Types..... | 3-6 |
| Variant Data Types..... | 3-7 |
| User-Defined Data Types..... | 3-8 |
| Data Type Conversions | 3-9 |

| | |
|---|------|
| Arrays..... | 3-10 |
| Declaring an Array | 3-10 |
| Referencing an Array..... | 3-10 |
| Dynamic Arrays..... | 3-11 |
| Dimensions of a Dynamic Array | 3-11 |
| Dynamic Array Example..... | 3-11 |
| Expressions and Operators..... | 3-12 |
| Numeric Operators..... | 3-12 |
| String Concatenation Operators..... | 3-12 |
| Comparison Operators | 3-13 |
| Logical Operators | 3-13 |
| Scope of Variables and Constants | 3-14 |
| Year 2000 Compliance | 3-15 |
| Suggestions for Avoiding Year 2000 Problems..... | 3-16 |
| Trappable Errors | 3-16 |
| Responding to Errors..... | 3-17 |
| User-Defined Errors..... | 3-17 |
| Error-Handling Examples..... | 3-17 |
| | |
| 4 SQABasic Scripts | |
| What is a Script? | 4-1 |
| Script Source Files | 4-2 |
| Script Executable Files | 4-2 |
| Script Structure | 4-2 |
| Sample Script | 4-4 |
| User Action and Verification Point Commands | 4-6 |
| User Action Commands..... | 4-6 |
| Verification Point Commands..... | 4-7 |
| Syntax of User Action and Verification Point Commands | 4-8 |
| Components of a Recognition Method String..... | 4-10 |
| Recognition Method Order | 4-10 |
| Recognition Methods in Java Commands..... | 4-13 |
| Object Context..... | 4-15 |
| Establishing Context through a Window Command..... | 4-15 |
| Establishing Context through Context Notation..... | 4-18 |
| Default Context..... | 4-20 |

Contents

| | |
|---|------|
| Customizing Scripts | 4-20 |
| Script Editing Basics..... | 4-21 |
| Declaring Variables and Constants | 4-21 |
| Adding Custom Procedures to a Script | 4-23 |
| Adding Custom Procedures to a Library File..... | 4-26 |
| Using SQABasic Header Files..... | 4-29 |
| Sample Library and Header Files..... | 4-32 |
| Using the Template File | 4-33 |
| | |
| 5 Enhancements to Recorded Scripts | |
| Object Scripting | 5-1 |
| Specifying an Object | 5-2 |
| Specifying the Object Property..... | 5-6 |
| Array of Property Values..... | 5-8 |
| Getting Help Defining Recognition Methods..... | 5-9 |
| Object Scripting Status Codes..... | 5-11 |
| Managing Custom Verification Points | 5-12 |
| Summary of Verification Point Management Commands | 5-13 |
| Current Baseline and Logged Baseline..... | 5-14 |
| Using the Verification Point Management Commands..... | 5-16 |
| Ownership of Custom Verification Point Files..... | 5-20 |
| Comparing Environment States | 5-21 |
| Why Compare Environment States? | 5-21 |
| What Environment State Changes Are Detected?..... | 5-21 |
| Using the Environment State Comparison Commands..... | 5-21 |
| Specifying the Areas of the Environment To Test..... | 5-23 |
| Example of an Environment State Comparison..... | 5-23 |
| Displaying Messages in Robot | 5-26 |
| Displaying Messages in the Console Window..... | 5-27 |
| Displaying Messages in the LogViewer | 5-28 |
| Using Datapools..... | 5-30 |
| Accessing External Applications..... | 5-31 |
| Dynamic Data Exchange (DDE) | 5-31 |
| Objects | 5-32 |

Part III Command Reference

| | |
|-------------------------------------|------|
| 6 Command Reference | |
| Abs | 6-2 |
| AnimateControl | 6-2 |
| AnimateControlVP..... | 6-4 |
| AppActivate..... | 6-6 |
| Asc | 6-7 |
| Assert | 6-7 |
| Atn | 6-8 |
| Beep..... | 6-8 |
| Begin Dialog..End Dialog | 6-9 |
| Browser | 6-13 |
| Button..... | 6-16 |
| ButtonGroup..... | 6-17 |
| Calendar | 6-18 |
| CalendarVP | 6-20 |
| Call | 6-21 |
| CallScript..... | 6-23 |
| CancelButton | 6-24 |
| Caption | 6-25 |
| CCur..... | 6-26 |
| CDBl..... | 6-27 |
| ChDir | 6-27 |
| ChDrive..... | 6-28 |
| CheckBox (Statement)..... | 6-29 |
| CheckBox (User Action Command)..... | 6-30 |
| CheckBoxVP | 6-32 |
| Chr..... | 6-35 |
| CInt | 6-36 |
| Class List..... | 6-37 |
| Clipboard..... | 6-37 |
| ClipboardVP..... | 6-38 |
| CLng..... | 6-39 |
| Close..... | 6-40 |

Contents

| | |
|--------------------------------------|------|
| ComboBox (Statement) | 6-41 |
| ComboBox (User Action Command) | 6-42 |
| ComboBoxVP | 6-44 |
| ComboEditBox | 6-48 |
| ComboEditBoxVP | 6-50 |
| ComboListBox | 6-52 |
| ComboListBoxVP | 6-55 |
| Command | 6-58 |
| Const | 6-59 |
| Cos | 6-60 |
| CreateObject | 6-61 |
| CSng | 6-62 |
| CStr | 6-63 |
| 'SCStrings | 6-64 |
| CurDir | 6-65 |
| CVar | 6-66 |
| CVDate | 6-66 |
| DataWindow | 6-68 |
| DataWindowVP | 6-72 |
| Date (Function) | 6-74 |
| Date (Statement) | 6-75 |
| DateSerial | 6-76 |
| DateTime | 6-77 |
| DateTimeVP | 6-78 |
| DateValue | 6-79 |
| Day | 6-80 |
| DDEAppReturnCode | 6-81 |
| DDEExecute | 6-82 |
| DDEInitiate | 6-83 |
| DDEPoke | 6-85 |
| DDERequest | 6-86 |
| DDETerminate | 6-88 |
| Declare | 6-89 |
| Def $type$ | 6-91 |
| DelayFor | 6-92 |

| | |
|-----------------------------------|-------|
| Desktop | 6-93 |
| Dialog (Function)..... | 6-94 |
| Dialog (Statement) | 6-96 |
| Dim..... | 6-97 |
| Dir | 6-100 |
| DlgControlID..... | 6-102 |
| DlgEnable (Function) | 6-104 |
| DlgEnable (Statement)..... | 6-105 |
| DlgEnd | 6-107 |
| DlgFocus (Function)..... | 6-109 |
| DlgFocus (Statement) | 6-110 |
| DlgListBoxArray (Function) | 6-111 |
| DlgListBoxArray (Statement) | 6-113 |
| DlgSetPicture | 6-114 |
| DlgText (Function) | 6-116 |
| DlgText (Statement) | 6-118 |
| DlgValue (Function) | 6-120 |
| DlgValue (Statement)..... | 6-122 |
| DlgVisible (Function) | 6-123 |
| DlgVisible (Statement)..... | 6-124 |
| Do...Loop..... | 6-126 |
| DoEvents | 6-127 |
| DropComboBox | 6-128 |
| DropListBox..... | 6-129 |
| EditBox..... | 6-131 |
| EditBoxVP..... | 6-133 |
| EndSaveWindowPositions | 6-136 |
| Environ..... | 6-137 |
| Eof..... | 6-138 |
| Erase | 6-139 |
| Erl | 6-141 |
| Err (Function) | 6-142 |
| Err (Statement)..... | 6-143 |
| Error (Function)..... | 6-144 |
| Error (Statement) | 6-145 |

Contents

| | |
|-------------------------------------|-------|
| Exit..... | 6-146 |
| Exp..... | 6-147 |
| FileAttr..... | 6-148 |
| FileCopy..... | 6-149 |
| FileDateTime..... | 6-150 |
| FileLen..... | 6-151 |
| FileVP..... | 6-152 |
| Fix..... | 6-153 |
| For...Next..... | 6-154 |
| Format..... | 6-156 |
| FreeFile..... | 6-164 |
| Function...End Function..... | 6-164 |
| FV..... | 6-166 |
| GenericObject..... | 6-168 |
| GenericObjectVP..... | 6-170 |
| Get..... | 6-173 |
| GetAttr..... | 6-175 |
| GetField..... | 6-176 |
| GetLastVPResult..... | 6-177 |
| GetObject..... | 6-177 |
| Global..... | 6-179 |
| GoTo..... | 6-182 |
| GroupBox (Statement)..... | 6-183 |
| GroupBox (User Action Command)..... | 6-184 |
| GroupBoxVP..... | 6-186 |
| Header..... | 6-188 |
| HeaderVP..... | 6-190 |
| Hex..... | 6-192 |
| HotKeyControl..... | 6-193 |
| HotKeyControlVP..... | 6-194 |
| Hour..... | 6-195 |
| HTML..... | 6-196 |
| HTMLVP..... | 6-198 |
| HTMLActiveX..... | 6-200 |
| HTMLActiveXVP..... | 6-201 |

| | |
|-------------------------|-------|
| HTMLDocument | 6-203 |
| HTMLDocumentVP | 6-204 |
| HTMLHiddenVP | 6-206 |
| HTMLImage | 6-208 |
| HTMLImageVP | 6-209 |
| HTMMLink | 6-211 |
| HTMMLinkVP | 6-212 |
| HTMLTable..... | 6-214 |
| HTMLTableVP..... | 6-215 |
| If...Then...Else | 6-217 |
| '\$Include | 6-218 |
| Input (Function)..... | 6-220 |
| Input (Statement) | 6-221 |
| InputBox..... | 6-222 |
| InputChars | 6-223 |
| InputKeys | 6-224 |
| InStr | 6-229 |
| Int | 6-230 |
| IPAddress..... | 6-232 |
| IPAddressVP..... | 6-233 |
| IPmt | 6-234 |
| IRR | 6-236 |
| Is | 6-237 |
| IsDate..... | 6-238 |
| IsEmpty | 6-238 |
| IsMissing | 6-239 |
| IsNull..... | 6-240 |
| IsNumeric | 6-241 |
| JavaCanvas | 6-242 |
| JavaCanvasVP | 6-244 |
| JavaListView | 6-246 |
| JavaListViewVP..... | 6-248 |
| JavaMenu..... | 6-250 |
| JavaMenuVP..... | 6-251 |
| JavaObject..... | 6-253 |

Contents

| | |
|-------------------------------------|-------|
| JavaObjectVP | 6-255 |
| JavaPanel..... | 6-256 |
| JavaPanelVP | 6-258 |
| JavaPopupMenu | 6-260 |
| JavaPopupMenuVP | 6-261 |
| JavaSplitPane | 6-263 |
| JavaSplitPaneVP | 6-264 |
| JavaSplitter..... | 6-266 |
| JavaSplitterVP | 6-268 |
| JavaTable | 6-270 |
| JavaTableVP..... | 6-272 |
| JavaTableHeader | 6-273 |
| JavaTableHeaderVP | 6-275 |
| JavaTree..... | 6-277 |
| JavaTreeVP..... | 6-279 |
| JavaWindow..... | 6-280 |
| JavaWindowVP | 6-282 |
| Kill | 6-284 |
| Label | 6-285 |
| LabelVP..... | 6-286 |
| LBound..... | 6-289 |
| LCase | 6-290 |
| Left..... | 6-290 |
| Len..... | 6-292 |
| Let..... | 6-292 |
| Like..... | 6-293 |
| Line Input..... | 6-294 |
| ListBox (Statement) | 6-296 |
| ListBox (User Action Command) | 6-297 |
| ListBoxVP..... | 6-300 |
| ListView | 6-303 |
| ListViewVP..... | 6-305 |
| Loc | 6-307 |
| Lock | 6-308 |
| Lof..... | 6-310 |

| | |
|--------------------------|-------|
| Log..... | 6-311 |
| Lset | 6-312 |
| LTrim | 6-313 |
| MenuIDSelect..... | 6-314 |
| MenuSelect..... | 6-314 |
| Mid (Function)..... | 6-316 |
| Mid (Statement) | 6-317 |
| Minute | 6-318 |
| MkDir..... | 6-319 |
| ModuleVP | 6-320 |
| Month..... | 6-322 |
| MsgBox (Function) | 6-322 |
| MsgBox (Statement) | 6-325 |
| Name | 6-326 |
| New | 6-327 |
| '\$NoCStrings..... | 6-328 |
| Nothing | 6-329 |
| Now..... | 6-330 |
| NPV | 6-330 |
| Null | 6-331 |
| Object Class | 6-332 |
| Oct..... | 6-334 |
| OKButton..... | 6-335 |
| On...GoTo..... | 6-336 |
| On Error | 6-337 |
| Open..... | 6-338 |
| Option Base | 6-340 |
| Option Compare | 6-342 |
| Option Explicit | 6-343 |
| OptionButton..... | 6-343 |
| OptionGroup..... | 6-344 |
| Pager | 6-346 |
| PagerVP | 6-347 |
| PasswordBox | 6-348 |
| Picture | 6-350 |

Contents

| | |
|--|-------|
| PlayJrnl | 6-351 |
| Pmt | 6-352 |
| PopupMenuIDSelect | 6-353 |
| PopupMenuSelect | 6-353 |
| PPmt | 6-355 |
| Print | 6-356 |
| ProgressBar | 6-358 |
| ProgressBarVP | 6-359 |
| PSGrid | 6-362 |
| PSGridHeader | 6-365 |
| PSGridHeaderVP | 6-366 |
| PSGridVP | 6-368 |
| PSMenu | 6-371 |
| PSMenuVP | 6-372 |
| PSNavigator | 6-373 |
| PSNavigatorVP | 6-375 |
| PSPanel | 6-378 |
| PSPanelVP | 6-382 |
| PSSpin | 6-386 |
| PSSpinVP | 6-388 |
| PSTree | 6-390 |
| PSTreeHeader | 6-392 |
| PSTreeHeaderVP | 6-394 |
| PSTreeVP | 6-396 |
| PushButton (Statement) | 6-399 |
| PushButton (User Action Command) | 6-400 |
| PushButtonVP | 6-402 |
| Put | 6-405 |
| PV | 6-406 |
| RadioButton | 6-407 |
| RadioButtonVP | 6-409 |
| Randomize | 6-412 |
| Rate | 6-413 |
| Rebar | 6-414 |
| RebarVP | 6-416 |

| | |
|------------------------|-------|
| ReDim | 6-417 |
| RegionVP | 6-419 |
| Rem | 6-421 |
| Reset | 6-422 |
| ResetTime | 6-423 |
| Resume | 6-423 |
| RichEdit | 6-424 |
| RichEditVP | 6-426 |
| Right | 6-428 |
| Rmdir | 6-429 |
| Rnd | 6-430 |
| Rset | 6-431 |
| RTrim | 6-432 |
| ScrollBar | 6-433 |
| ScrollBarVP | 6-435 |
| Second | 6-437 |
| Seek (Function) | 6-438 |
| Seek (Statement) | 6-439 |
| Select Case | 6-441 |
| SendKeys | 6-443 |
| Set | 6-443 |
| SetAttr | 6-444 |
| SetField | 6-446 |
| SetThinkAvg | 6-447 |
| SetTime | 6-448 |
| Sgn | 6-448 |
| Shell | 6-449 |
| Sin | 6-450 |
| Space | 6-451 |
| Spc | 6-452 |
| SpinControl | 6-453 |
| SpinControlVP | 6-454 |
| SQAConsoleClear | 6-456 |
| SQAConsoleWrite | 6-456 |
| SQADatapoolClose | 6-457 |

Contents

| | |
|----------------------------------|-------|
| SQADatapoolFetch | 6-458 |
| SQADatapoolOpen | 6-459 |
| SQADatapoolRewind..... | 6-462 |
| SQADatapoolValue | 6-463 |
| SQAEEnvCreateBaseline | 6-465 |
| SQAEEnvCreateCurrent..... | 6-466 |
| SQAEEnvCreateDelta | 6-467 |
| SQAFindObject | 6-469 |
| SQAGetCaptionTerminatorChar..... | 6-470 |
| SQAGetChildren | 6-471 |
| SQAGetDir | 6-473 |
| SQAGetLogDir | 6-474 |
| SQAGetOcrRegionRect..... | 6-474 |
| SQAGetOcrRegionText | 6-476 |
| SQAGetProperty | 6-478 |
| SQAGetPropertyArray..... | 6-480 |
| SQAGetPropertyArrayAsString..... | 6-482 |
| SQAGetPropertyArraySize | 6-483 |
| SQAGetPropertyAsString..... | 6-485 |
| SQAGetPropertyNames | 6-487 |
| SQAGetSystemLong..... | 6-488 |
| SQAINvokeMethod..... | 6-490 |
| SQALogMessage | 6-492 |
| SQAQueryKey | 6-493 |
| SQAResumeLogOutput..... | 6-493 |
| SQAScriptCmdFailure..... | 6-494 |
| SQASetAssignmentChar..... | 6-494 |
| SQASetCaptionTerminatorChar..... | 6-495 |
| SQASetDefaultBrowser | 6-496 |
| SQASetProperty | 6-498 |
| SQASetSeparatorChar | 6-500 |
| SQAShellExecute | 6-501 |
| SQASuspendLogOutput..... | 6-502 |
| SQASyncPointWait..... | 6-502 |
| SQAVpGetActualFileName..... | 6-503 |

| | |
|---------------------------------------|-------|
| SQAVpGetBaselineFileName..... | 6-504 |
| SQAVpGetCurrentBaselineFileName | 6-505 |
| SQAVpLog | 6-506 |
| SQAWaitForObject..... | 6-508 |
| SQAWaitForPropertyValue | 6-509 |
| SQLClose | 6-511 |
| SQLError..... | 6-512 |
| SQLExecQuery | 6-513 |
| SQLGetSchema..... | 6-514 |
| SQLOpen | 6-516 |
| SQLRequest | 6-517 |
| SQLRetrieve..... | 6-518 |
| SQLRetrieveToFile..... | 6-520 |
| Sqr | 6-521 |
| StartApplication | 6-522 |
| StartBrowser | 6-523 |
| StartJavaApplication..... | 6-524 |
| StartSaveWindowPositions | 6-526 |
| StartTimer | 6-527 |
| Static | 6-527 |
| StaticComboBox | 6-529 |
| StatusBar..... | 6-530 |
| StatusBarVP..... | 6-532 |
| Stop..... | 6-534 |
| StopTimer | 6-534 |
| Str | 6-535 |
| StrComp..... | 6-535 |
| String | 6-537 |
| Sub...End Sub..... | 6-538 |
| SysMenuIDSelect..... | 6-539 |
| SysMenuSelect | 6-540 |
| Tab..... | 6-541 |
| TabControl..... | 6-542 |
| TabControlVP..... | 6-544 |
| Tan..... | 6-547 |

Contents

| | |
|-----------------------|-------|
| Text..... | 6-548 |
| TextBox..... | 6-549 |
| Time (Function)..... | 6-550 |
| Time (Statement)..... | 6-551 |
| Timer..... | 6-552 |
| TimeSerial..... | 6-553 |
| TimeValue..... | 6-554 |
| Toolbar..... | 6-555 |
| ToolbarVP..... | 6-556 |
| Trackbar..... | 6-559 |
| TrackbarVP..... | 6-561 |
| TreeView..... | 6-563 |
| TreeViewVP..... | 6-567 |
| Trim..... | 6-570 |
| Type..... | 6-570 |
| Typeof..... | 6-572 |
| TypingDelays..... | 6-572 |
| UBound..... | 6-573 |
| UCase..... | 6-575 |
| Unlock..... | 6-575 |
| Val..... | 6-577 |
| VarType..... | 6-578 |
| WebSiteVP..... | 6-579 |
| Weekday..... | 6-581 |
| While...Wend..... | 6-582 |
| Width..... | 6-584 |
| Window..... | 6-585 |
| WindowVP..... | 6-591 |
| With..... | 6-594 |
| Write..... | 6-595 |
| Year..... | 6-597 |

Appendixes

- A SQABasic Syntax Summary
- B Trappable Error Codes
- C Object Scripting Status Codes
- D Derived Trigonometric Functions
- E Mouse Actions

Glossary

Index

Contents

▶ ▶ ▶ Preface

Welcome to the *SQABasic Language Reference*. The *SQABasic Language Reference* describes the commands and conventions of the SQABasic scripting language. SQABasic includes most of the syntax rules and core commands found in the industry-standard Microsoft® Basic language.

Audience

This guide is intended to help QA managers, developers, and test engineers read and customize scripts generated with Rational Robot. Familiarity with Robot and other Rational Test software is assumed. Familiarity with programming language practices (but not necessarily Microsoft Basic programming) is also assumed.

Other Resources

- ▶ This product contains complete online Help. For information about calling SQABasic Help, see the following section.
- ▶ All manuals are available online in PDF format. The online manuals are on the Rational solutions for Windows Online Documentation CD.
- ▶ For information about training opportunities, see the Rational University Web site: <http://www.rational.com/university>.

Accessing SQABasic Help

You can access SQABasic Help in a variety of ways:

- ▶ From the Start menu, click SQABasic Language Reference in the installation directory of your Rational product (typically, Rational Test).
- ▶ From within Robot, click Help → SQABasic Reference.

Accessing SQABasic Help

- ▶ While you are editing a script in Robot, you can display context-sensitive information about a particular SQABasic command. To do so:
 1. Place the insertion point immediately before, after, or anywhere within the command name.
 2. Press F1.

If a single Help topic is associated with the command name, reference information about that command appears immediately.

If multiple Help topics are associated with the command, the topics are listed in the Topics Found dialog box. Select the topic you want and click Display.

Using the Examples in Help

The Help system offers a small working example or code fragment of most SQABasic commands. To see an example of a command, click the word Example under the command name.

Clicking on Example opens a separate window containing a code example. You can simply look at the contents of this window, or you can copy the example into a Robot script.

To copy an example into a script, follow these steps:

1. In Robot, click File → Open Script.
2. Type or select a script name and click OK.
3. In the SQABasic Help Example window, click the Copy button to copy the example to the Clipboard.

To copy part of the example, select the text you want to copy and press CTRL+C.

4. Paste the contents into the Robot window. (If you copy the whole example, delete the lines of description that appear before the example.)

Notes About the Examples

- ▶ To run the examples that show ODBC commands (those beginning with SQL), you need to have Microsoft Access® installed on your machine.
- ▶ To run the examples that show Object commands, you need to have VISIO® installed on your machine.
- ▶ Some commands do not have examples associated with them.

Typographical Conventions

This manual uses the following typographical conventions:

| Convention | Where used |
|--|---|
| Monospace type | SQABasic keywords and examples: Use the <code>Dim</code> statement to... <code>Dim i As Integer</code> |
| Monospace type with the first letter of key syllables uppercase | SQABasic commands: <code>Len(string\$)</code> <code>DateValue(string\$)</code> |
| <i>Italicized monospace type</i> | Arguments to commands: <code>Rmdir string\$</code> <code>CVar (expression)</code> |
| <i>Italicized variables and/or type-declaration characters in brackets</i> | Optional arguments: [, <i>caption</i> \$] [<i>type</i> \$] [\$] |
| A list inside braces ({ }) with a vertical bar () separating choices | A list of required choices for an argument. One choice must be selected: {Goto <i>label</i> Resume Next Goto 0} |
| Bold type | New terms: An array has one or more dimensions . |
| <i>Italicized type</i> | Emphasized text, manual titles, and section headings: ...is assumed to be the <i>current context</i> . See <i>Dialog Box Commands</i> on page 1-19. |
| An arrow (→) between menu commands | Between a menu or sub-menu name and a menu command. Choose each command in sequence. For example, File → Save As means click File from the menu bar, and then click Save As from the menu. |
| The symbol ► ► ► before or after a table | Indicates a table is continued on the next page or continued from the previous page. |

Contacting Rational Technical Publications

Contacting Rational Technical Publications

To send feedback about documentation for Rational products, please send e-mail to our technical publications department at techpubs@rational.com.

Contacting Rational Technical Support

If you have questions about installing, using, or maintaining this product, contact Rational Technical Support as follows:

| Location | Contact Information | Notes |
|----------------|---|---|
| North America | Telephone: 800-433-5444 408-863-4000 E-mail: support@europe.rational.com | Please be prepared to supply the following information: <ul style="list-style-type: none">▶ Your name, telephone number, and company name |
| Europe | Telephone: + 31 (0) 20 4546 200 E-mail: support@europe.rational.com | <ul style="list-style-type: none">▶ Computer make and model▶ Operating system and version number |
| Asia Pacific | Telephone: + 61-2-9419-0111 E-mail: support@apac.rational.com | <ul style="list-style-type: none">▶ Product release number and serial number▶ Your Case ID number (if you are calling about a previously reported problem) |
| World Wide Web | http://www.rational.com | Click the Technical Support link. |

▶ ▶ ▶ Part I

Introducing SQABasic

▶ ▶ ▶ C H A P T E R 1

What Is SQABasic?

SQABasic is the Rational Software Corporation language for building GUI scripts.

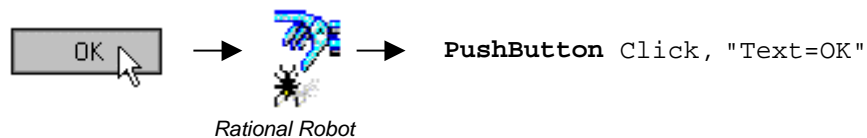
SQABasic includes most of the syntax rules and core commands found in the industry-standard Microsoft Basic language. If you're familiar with Microsoft Basic or Visual Basic, you're already familiar with much of the SQABasic language.

Along with support for Basic commands, SQABasic includes command **additions** — commands specifically designed for use in Rational TestStudio and Rational PerformanceStudio scripts.

Automatic Script Generation

Generating an SQABasic script might be the briefest development experience you'll ever have. That's because Rational Robot automatically generates a script for you when you record the script.

During GUI recording, Robot "watches" every keyboard and mouse action you take in the application-under-test. Robot translates these actions into a series of SQABasic commands and stores them in the script. For example, when you click an OK button, Robot represents the action as `PushButton Click, "Text=OK"`:



When you finish recording, you can play it back immediately. Robot compiles the script before beginning to play it back.

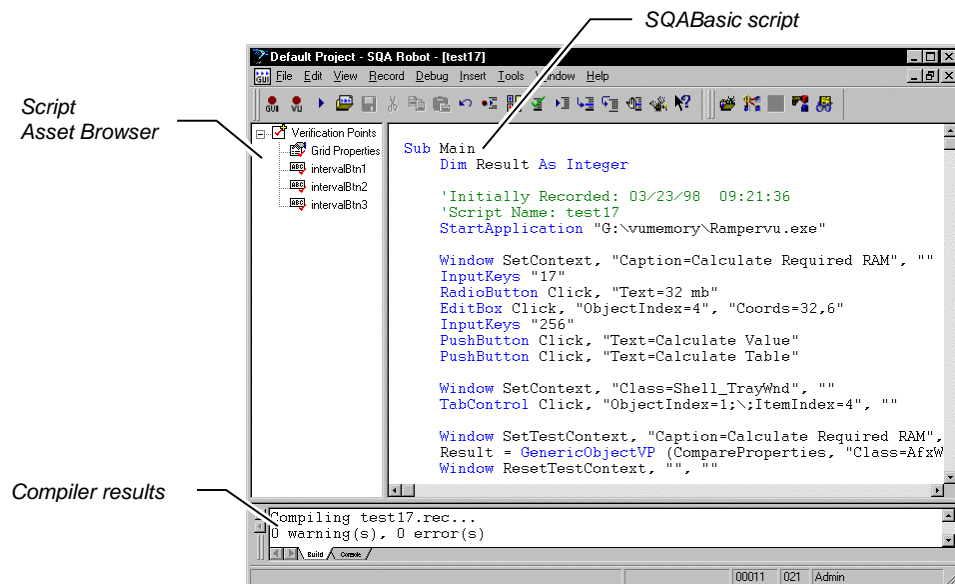
Working with Scripts

Although Robot generates complete, executable scripts, sometimes you might want to edit a recorded script — for example, to:

- ▶ Add Do...While or For...Next loops to simplify repetitive actions
- ▶ Add conditional branching
- ▶ Perform Object Scripting functions
- ▶ Add datapool commands
- ▶ Access OLE or DDE resources
- ▶ Request user input during script playback, or display a message box to report some unusual event during playback
- ▶ Perform a variety of math, date, and time functions
- ▶ Respond to runtime errors

Your Work Environment

With SQABasic as your scripting language, you view, edit, compile, debug, and run scripts through Robot. Here is an example of the Robot environment:



For information about Robot, see the *Using Rational Robot* manual.

Source and Runtime Files

SQABasic supports the following kinds of files:

| File type | Extension |
|----------------------------------|------------|
| Script file | .rec |
| Header file | .sbh |
| Library source file | .sbl, .rec |
| Script and library runtime files | .sbx |

SQABasic Additions to the Basic Language

SQABasic provides a number of commands in addition to the commands in the Microsoft Basic language. The following categories of commands are provided to help you test your applications and analyze the results:

Datapool Commands – Control access to a datapool. You can use a datapool to supply values to scripts during playback. You create datapools with TestManager.

Object Scripting commands – Access an application's objects and object properties from within a script. Object scripting tasks include retrieving and setting an object's properties. Object Scripting commands can only be added to a script manually during editing. Robot does not generate these commands.

Timing and Coordination Commands – Time user activities and control the rate of script playback.

User Action commands – Perform user actions on specific objects while recording. Actions include choosing a menu command, scrolling a list box, clicking a button, or typing text into an edit box.

Utility commands – Perform a variety of actions such as calling other scripts, playing back low-level recordings, controlling output to the LogViewer or Robot console, and managing custom verification points.

Verification Point commands – Compare the results of a user action captured during playback against the results of the same action captured during recording. If the playback result matches the **recorded baseline** (the information captured during recording), the verification point passes. If the result is different, the verification point fails.

For a listing and brief description of the commands in each category, see Chapter 2, *Functional List*.

Other Commands Not Found in Basic

In addition to the above command categories, SQABasic provides these commands not found in standard Basic:

| | |
|------------|----------------|
| Assert | ' \$CStrings |
| GetField\$ | ' \$Include |
| SetField\$ | ' \$NoCStrings |

All SQABasic commands are described in Chapter 6, *Command Reference*.

VU Scripting Language

Because the SQABasic scripting language lets you capture keyboard and mouse actions as well as verify GUI objects, it is the language used in functional testing (testing the way your application looks and works).

But for testing client/server performance, you need to record a client's requests to the server. Capturing a client/server conversation requires the VU scripting language.

VU is a C-based language that Robot generates when recording requests such as HTTP, SQL, TUXEDO, and socket-level requests. Rational PerformanceStudio is required to record and run VU scripts. For information, see the *Using Rational LoadTest* manual.

▶ ▶ ▶ C H A P T E R 2

Functional List

This chapter organizes the SQABasic commands into functional categories.

NOTE: The SQABasic command category Web commands (HTTP and HTTP/HTTPS API requests to a server) is no longer supported in SQABasic. However, actions on HTML objects are supported in User Action commands and Verification Point commands. Also, Virtual User commands (with the exception of SQASyncPointWait) are no longer supported in SQABasic. You will find commands that perform similar functions in Rational Software's VU language.

Arrays

| | |
|-------------|---|
| Erase | Reinitialize the contents of an array. |
| LBound | Return the lower bound of an array subscript. |
| Option Base | Declare the default lower bound for array subscripts. |
| ReDim | Declare dynamic arrays and reallocate memory. |
| UBound | Return the upper bound of an array subscript. |

Compiler Directives

| | |
|--------------------------------------|--|
| '\$CStrings (SQABasic addition) | Treat a backslash in a string as an escape character as in the C language. |
| '\$Include (SQABasic addition) | Tell the compiler to include statements from another file. |
| '\$NoCStrings (SQABasic addition) | Tell the compiler to treat a backslash as a normal character. |
| Rem | Treat the remainder of the line as a comment. Equivalent to an apostrophe ('). |

Datapool Commands (SQABasic Additions)

These commands let you access data in a datapool.

| | |
|-------------------|--|
| SQADatapoolClose | Close the specified datapool. |
| SQADatapoolFetch | Move the cursor for the datapool to the next row. |
| SQADatapoolOpen | Open the specified datapool. |
| SQADatapoolRewind | Reset the cursor for the specified datapool. |
| SQADatapoolValue | Retrieve the value of the specified datapool column. |

Dates & Times

| | |
|----------------|---|
| Date Function | Return the current date. |
| Date Statement | Set the system date. |
| DateSerial | Return the date value for year, month, and day specified. |
| DateValue | Return the date value for string specified. |
| Day | Return the day of month component of a date-time value. |
| Hour | Return the hour of day component of a date-time value. |
| IsDate | Determine whether a value is a legal date. |
| Minute | Return the minute component of a date-time value. |
| Month | Return the month component of a date-time value. |
| Now | Return the current date and time. |
| Second | Return the second component of a date-time value. |
| Time Function | Return the current time. |
| Time Statement | Set the current time. |
| Timer | Return the number of seconds since midnight. |
| TimeSerial | Return the time value for hour, minute, and second. |
| TimeValue | Return the time value for string specified. |
| Weekday | Return the day of the week for the specified date-time. |
| Year | Return the year component of a date-time value. |

Declarations

| | |
|---------|--|
| Const | Declare a symbolic constant. |
| Declare | Forward declare a procedure in the same module or in a dynamic link library. |

| | |
|---------------------------|--|
| <i>DefType</i> | Declare the default data type for variables. |
| Dim | Declare variables. |
| Function ... End Function | Define a function. |
| Global | Declare a global variable. |
| Option Compare | Declare the default case sensitivity for string comparisons. |
| Option Explicit | Force all variables to be explicitly declared. |
| ReDim | Declare dynamic arrays and reallocate memory. |
| Static | Define a static variable or subprogram. |
| Sub ... End Sub | Define a subprogram. |
| Type | Declare a user-defined data type. |

Dialog Box Definition

| | |
|---------------------------------|--|
| Begin Dialog | Begin a dialog box definition. |
| Button | Define a button dialog box control. |
| ButtonGroup | Begin definition of a group of button dialog box controls. |
| CancelButton | Define a Cancel button dialog box control. |
| Caption | Define the title of a dialog box. |
| CheckBox | Define a check box dialog box control. |
| ComboBox | Define a combo box dialog box control. |
| DropComboBox | Define a drop-down combo box dialog box control. |
| DropListBox | Define a drop-down list box dialog box control. |
| GroupBox (SQABasic Addition) | Define a group box dialog box control. |
| ListBox | Define a list box dialog box control. |
| OKButton | Define an OK button dialog box control. |
| OptionButton | Define an option button dialog box control. |
| OptionGroup | Begin definition of a group of option button controls. |
| Picture | Define a Picture control. |
| PushButton | Define a push button dialog box control. |
| StaticComboBox | Define a static combo box dialog box control. |
| Text | Define a line of text in a dialog box. |
| TextBox | Define a text box in a dialog box. |

Dialog Box Services

| | |
|---------------------------|---|
| Dialog Function | Display a dialog box and return the button pressed. |
| Dialog Statement | Display a dialog box. |
| DlgControlID | Return the numeric ID of a dialog control. |
| DlgEnable Function | Tell whether a dialog control is enabled or disabled. |
| DlgEnable Statement | Enable or disable a dialog control. |
| DlgEnd | Close the active dialog box. |
| DlgFocus Function | Return the ID of the dialog control having input focus. |
| DlgFocus Statement | Set focus to a dialog control. |
| DlgListBoxArray Function | Return the contents of a list box or combo box. |
| DlgListBoxArray Statement | Set the contents of a list box or combo box. |
| DlgSetPicture | Change the picture in the Picture control. |
| DlgText Function | Return the text associated with a dialog control. |
| DlgText Statement | Set the text associated with a dialog control. |
| DlgValue Function | Return the value associated with dialog control. |
| DlgValue Statement | Set the value associated with a dialog control. |
| DlgVisible Function | Tell whether a control is visible or hidden. |
| DlgVisible Statement | Show or hide a dialog control. |

Disk and Directory Control

| | |
|---------|---|
| ChDir | Change the default directory for a drive. |
| ChDrive | Change the default drive. |
| CurDir | Return the current directory for a drive. |
| Dir | Return a filename that matches a pattern. |
| MkDir | Make a directory on a disk. |
| Rmdir | Remove a directory from a disk. |

Dynamic Data Exchange (DDE)

| | |
|------------------|---|
| DDEAppReturnCode | Return a code from an application on a DDE channel. |
| DDEExecute | Send commands to an application on a DDE channel. |
| DDEInitiate | Open a dynamic data exchange (DDE) channel. |
| DDEPoke | Send data to an application on a DDE channel. |
| DDERequest | Return data from an application on a DDE channel. |
| DDETerminate | Close a DDE channel. |

Environmental Control

| | |
|----------------|--|
| AppActivate | Activate another application. |
| Command | Return the command line by the MAIN sub. |
| Date Statement | Set the current date. |
| DoEvents | Let the operating system process messages. |
| Environ | Return a string from the operating system's environment. |
| Randomize | Initialize the random-number generator. |
| Shell | Run an executable program. |

Error Handling

| | |
|-------------------------------|--|
| Assert (SQABasic addition) | Trigger an error if a condition is false. |
| Erl | Return the line number where a runtime error occurred. |
| Err Function | Return a runtime error code. |
| Err Statement | Set the runtime error code. |
| Error Function | Return a string representing an error. |
| Error Statement | Generate an error condition. |
| On Error | Control runtime error handling. |
| Resume | End an error-handling routine. |

See Appendix B for a list of SQABasic trappable error codes.

File Control

| | |
|--------------|--|
| FileAttr | Return information about an open file. |
| FileCopy | Copy the contents of a file. |
| FileDateTime | Return modification date and time of a specified file. |
| FileLen | Return the length of specified file in bytes. |
| GetAttr | Return the attributes of a file, directory, or volume label. |
| Kill | Delete files from a disk. |
| Name | Rename a disk file. |
| SetAttr | Set attribute information for a file. |

File Input/Output

| | |
|-----------------|--|
| Close | Close a file. |
| Eof | Check for end of file. |
| FreeFile | Return the next unused file number. |
| Get | Read bytes from a file. |
| Input Function | Return a string of characters from a file. |
| Input Statement | Read data from a file or from the keyboard. |
| Line Input | Read a line from a sequential file or from the keyboard. |
| Loc | Return the current position of an open file. |
| Lock | Keep other processes from accessing part or all of an open file. |
| Lof | Return the length of an open file. |
| Open | Open a disk file or device for I/O. |
| Print | Print data to a file or to the screen. |
| Put | Write data to an open file. |
| Reset | Close all open disk files. |
| Seek Function | Return the current position for a file. |
| Seek Statement | Set the current position for a file. |
| Spc | Output a given number of spaces. |
| Tab | Move the print position to the given column. |
| Unlock | Restore access to an open file (release the lock). |
| Width | Set output-line width for an open file. |
| Write | Write data to a sequential file. |

Financial Functions

| | |
|------|--|
| FV | Return the future value for a stream of periodic cash flows. |
| IPmt | Return interest payment for a given period. |
| IRR | Return internal rate of return for a cash flow stream. |
| NPV | Return net present value of a cash flow stream. |
| Pmt | Return a constant payment per period for an annuity. |
| PPmt | Return principal payment for a given period. |
| PV | Return present value of a future stream of cash flows. |
| Rate | Return interest rate per period. |

Flow Control

| | |
|------------------|--|
| Call | Transfer control to a subprogram. |
| Do...Loop | Control repetitive actions. |
| Exit | Cause the current procedure or loop structure to return. |
| For...Next | Loop a fixed number of times. |
| GoTo | Send control to a line label. |
| If...Then...Else | Branch on a conditional value. |
| Let | Assign a value to a variable. |
| Lset | Left-align one string or a user-defined variable within another. |
| On...GoTo | Branch to one of several labels depending upon value. |
| Rset | Right-align one string within another. |
| Select Case | Execute one of a series of statement blocks. |
| Set | Set an object variable to a value. |
| Stop | Stop program execution. |
| While...Wend | Control repetitive actions. |
| With | Execute a series of statements on a specified variable. |

Numeric and Trigonometric Functions

| | |
|-----------|---|
| Abs | Return the absolute value of a number. |
| Atn | Return the arc tangent of a number. |
| Cos | Return the cosine of an angle. |
| Exp | Return the value of e raised to a power. |
| Fix | Return the integer part of a number. |
| Int | Return the integer part of a number. |
| IsNumeric | Determine whether a value is a legal number. |
| Log | Return the natural logarithm of a value. |
| Rnd | Return a random number. |
| Sgn | Return a value indicating the sign of a number. |
| Sin | Return the sine of an angle. |
| Sqr | Return the square root of a number. |
| Tan | Return the tangent of an angle. |

See Appendix D for a list of math functions derived from SQABasic Numeric and Trigonometric functions.

Object Scripting Commands (SQABasic Additions)

These commands let you work with an object's properties. The Object Scripting commands can only be used programmatically. Robot does not generate these commands during recording.

| | |
|-----------------------------|---|
| SQAFindObject | Search for a specified object. |
| SQAGetChildren | Retrieve an array containing recognition methods that identify each of an object's child objects. |
| SQAGetProperty | Retrieve the value of the specified property. |
| SQAGetPropertyArray | Retrieve an array of values for the specified property. |
| SQAGetPropertyArrayAsString | Retrieve the string equivalent of an array of values for the specified property. |
| SQAGetPropertyArraySize | Retrieve the number of elements in an array of property values. |
| SQAGetPropertyAsString | Retrieve a property value in <code>String</code> form. |
| SQAGetPropertyNames | Retrieve an array containing the names of all the object's properties. |

| | |
|-------------------------|---|
| SQAINvokeMethod | Execute the specified method of an object. |
| SQASetProperty | Assign a value to a specified property. |
| SQAWaitForObject | Pause execution of the script until the specified object can be found. |
| SQAWaitForPropertyValue | Pause execution of the script until a property is set to the specified value. |

Objects

| | |
|--------------|--|
| Class List | List of available classes. |
| Clipboard | Access the Windows Clipboard. |
| CreateObject | Create an OLE2 automation object. |
| GetObject | Retrieve an OLE2 object from a file or get the active OLE2 object for an OLE2 class. |
| Is | Determine if two object variables refer to the same object. |
| New | Allocate and initialize a new OLE2 object. |
| Nothing | Set an object variable to not refer to an object. |
| Object Class | Declare an OLE2 automation object. |
| Typeof | Check the class of an object. |
| With | Execute statements on an object or a user-defined type. |

ODBC

| | |
|-------------------|--|
| SQLClose | Close a data source connection. |
| SQLERROR | Return a detailed error message for ODBC functions. |
| SQLExecQuery | Execute an SQL statement. |
| SQLGetSchema | Obtain information about data sources, databases, terminology, users, owners, tables, and columns. |
| SQLOpen | Connect to a data source for use by other functions. |
| SQLRequest | Make a connection to a data source, execute an SQL statement, and return the results. |
| SQLRetrieve | Return the results of a select that was executed by SQLExecQuery into a user-provided array. |
| SQLRetrieveToFile | Return the results of a select that was executed by SQLExecQuery into a user-specified file. |

Screen Input/Output

| | |
|------------------|---|
| Beep | Produce a short beeping tone through the speaker. |
| Input Function | Return a string of characters from a file. |
| Input Statement | Read data from a file or from the keyboard. |
| InputDialog | Display a dialog box that prompts for input. |
| MsgBox Function | Display a Windows message box and return a value indicating which button the user selected. |
| MsgBox Statement | Display a prompt in a Windows message box. |
| PasswordBox | Display a dialog box for input. Do not echo input. |
| Print | Print data to a file or to the screen. |

SQABasic Commands

Most SQABasic additions to the Basic language are grouped within the following categories of commands:

- ▶ Datapool Commands. See page 2-2.
- ▶ Object Scripting Commands. See page 2-8.
- ▶ Timing and Coordination Commands. See page 2-12.
- ▶ User Action Commands. See page 2-12.
- ▶ Utility Commands. See page 2-14.
- ▶ Verification Point Commands. See page 2-17.

String Conversions

| | |
|------|--|
| Asc | Return an integer corresponding to a character code. |
| CCur | Convert a value to currency. |
| CDbl | Convert a value to double-precision floating point. |
| Chr | Convert a character code to a string. |
| CInt | Convert a value to an integer by rounding. |
| CLng | Convert a value to a long by rounding. |

| | |
|--------|---|
| CSng | Convert a value to single-precision floating point. |
| CStr | Convert a value to a string. |
| CVar | Convert a number or string to a variant. |
| CVDate | Convert a value to a variant date. |
| Format | Convert a value to a string using a picture format. |
| Val | Convert a string to a number. |

String Manipulation

| | |
|---------------------------------|---|
| GetField (SQABasic addition) | Return a substring from a delimited source string. |
| Hex | Return the hexadecimal representation of a number, as a string. |
| InStr | Return the position of one string within another. |
| LCase | Convert a string to lowercase. |
| Left | Return the left portion of a string. |
| Len | Return the length of a string or size of a variable. |
| Like Operator | Compare a string against a pattern. |
| LTrim | Remove leading spaces from a string. |
| Mid Function | Return a portion of a string. |
| Mid Statement | Replace a portion of a string with another string. |
| Oct | Return the octal representation of a number, as a string. |
| Right | Return the right portion of a string. |
| RTrim | Remove trailing spaces from a string. |
| SetField (SQABasic addition) | Replace a substring within a delimited target string. |
| Space | Return a string of spaces. |
| Str | Return the string representation of a number. |
| StrComp | Compare two strings. |
| String | Return a string consisting of a repeated character. |
| Trim | Remove leading and trailing spaces from a string. |
| UCase | Convert a string to uppercase. |

Timing and Coordination Commands (SQABasic Additions)

These commands affect the flow of test procedure playback by setting wait times and starting and stopping timers:

| | |
|------------------|---|
| DelayFor | Delay execution of the script for a specified number of milliseconds. |
| ResetTime | Reset the delay between execution of script commands to the default delay. |
| SetThinkAvg | Set the average “think time” for the next Robot action. |
| SetTime | Set the delay between script commands to the specified number of millisecond. |
| SQASyncPointWait | Define a sync point for coordination in multi-user testing. |
| StartTimer | Start the specified timer in the currently running script and write a message to the log. |
| StopTimer | Stop the specified timer in the currently running script and write the elapsed time in milliseconds to the log. |
| TypingDelays | Set one or more keystroke delays during playback of the next <code>InputKeys</code> command. |

User Action Commands (SQABasic Additions)

These commands cause an action to be taken on a particular control. Actions include choosing a menu command, scrolling a list box, clicking on a button, or typing text in an edit box:

| | |
|----------------|--|
| AnimateControl | Perform an action on an animation control. |
| Calendar | Perform an action on a month calendar control. |
| CheckBox | Perform an action on a check box control. |
| ComboBox | Perform an action on a combo box control. |
| ComboEditBox | Perform an action on a combo edit box control. |
| ComboListBox | Perform an action on a combo list box control. |
| DataWindow | Perform an action on a PowerBuilder DataWindow. |
| DateTime | Perform an action on a date and time (DTP) picker control. |
| Desktop | Perform an action on the Windows desktop. |
| EditBox | Perform an action on an edit box control. |
| GenericObject | Perform an action on a generic object. |
| GroupBox | Perform an action on a group box control. |

User Action Commands (SQABasic Additions)

| | |
|-------------------|---|
| Header | Perform an action on a header control. |
| HotKeyControl | Perform an action on a hot key control. |
| HTML | Perform a mouse action on an HTML tag. |
| HTMLActiveX | Perform a mouse action on an ActiveX control embedded in the page. |
| HTMLDocument | Perform a mouse click on the text of a Web page. |
| HTMLImage | Perform a mouse click on an image in a Web page. |
| HTMLLink | Perform a mouse click on link of a Web page. |
| HTMLTable | Perform a mouse click on a table in a Web page. |
| InputChars | Send one or more characters to the active window as if they had been entered at the keyboard. |
| InputKeys | Send one or more keystrokes to the active window as if they had been entered at the keyboard. |
| IPAddress | Perform an action on an IP Address control. |
| JavaCanvas | Perform an action on a Java canvas component. |
| JavaListView | Perform an action on a Java multi-column list component. |
| JavaMenu | Perform an action on a Java menu. |
| JavaObject | Perform an action on an unrecognized Java component. |
| JavaPanel | Perform an action on a Java panel or canvas. |
| JavaPopupMenu | Perform an action on a Java popup menu. |
| JavaSplitPane | Perform an action on a Java split pane. |
| JavaSplitter | Perform an action on a Java splitter. |
| JavaTable | Perform an action on a Java table. |
| JavaTableHeader | Perform an action on a Java table header. |
| JavaTree | Perform an action on a Java tree component. |
| JavaWindow | Perform an action on a Java window. |
| Label | Perform an action on a label control. |
| ListBox | Perform an action on a list box control. |
| ListView | Perform an action on a list view control. |
| MenuIDSelect | Perform a menu selection based on the internal ID of the menu item. |
| MenuSelect | Select a popup item through one or more mouse clicks. |
| Pager | Perform an action on a Pager control. |
| PopupMenuIDSelect | Perform a popup menu selection based on the internal ID of the menu item. |
| PopupMenuSelect | Select a popup menu item through one or more mouse clicks. |
| ProgressBar | Perform an action on a progress bar control. |

Utility Commands (SQABasic Additions)

| | |
|-----------------|--|
| PSGrid | Perform an action on a PeopleTools grid. |
| PSGridHeader | Perform an action on a column header in a PeopleTools grid. |
| PSMenu | Perform an action on a PeopleTools menu object. |
| PSNavigator | Perform an action on a PeopleTools Navigator window or a navigator map in the PeopleTools Business Process Designer. |
| PSPanel | Perform an action on a PeopleTools panel. |
| PSSpin | Perform an action on a PeopleTools spin control. |
| PSTree | Perform an action on a PeopleTools tree object. |
| PSTreeHeader | Perform an action on a column header in a PeopleTools tree object. |
| PushButton | Perform an action on a push button control. |
| RadioButton | Perform an action on an option button control. |
| Rebar | Perform an action on a Rebar control. |
| RichEdit | Perform an action on a rich edit control. |
| ScrollBar | Perform an action on a scroll bar control. |
| SpinControl | Perform an action on a spin control. |
| StatusBar | Perform an action on a status bar control. |
| SysMenuIDSelect | Perform a system menu selection based on the internal ID of the menu item. |
| SysMenuSelect | Perform a system menu selection based on the text of the menu item. |
| TabControl | Perform an action on a tab control. |
| ToolBar | Perform an action on a toolbar control. |
| Trackbar | Perform an action on a trackbar control. |
| TreeView | Perform an action on a treeview control. |
| Window | Perform an action on a window. |

Utility Commands (SQABasic Additions)

These commands affect the flow of script playback by setting wait times, calling other scripts, starting applications, starting and stopping timers, and playing back low-level recordings. They also control output to the log, retrieve results from running scripts, and set characters used in SQABasic statements:

| | |
|------------|---|
| Browser | Perform an action on a Web browser. |
| CallScript | Cause a script to be executed from within the currently-running script. |

Utility Commands (SQABasic Additions)

| | |
|-----------------------------|--|
| DelayFor | Delay execution of the script for a specified number of milliseconds. |
| EndSaveWindowPositions | Mark the end of the script commands that save the window positions for restoration at playback. |
| GetLastVPRResult | Return the result of the last verification point to have been evaluated in the current playback session. |
| PlayJrnl | Start playback of a series of low-level recorded mouse and keyboard actions. |
| ResetTime | Reset the delay between execution of script commands to the default delay. |
| SetTime | Set the delay between script commands to the specified number of millisecond. |
| SQAConsoleClear | Clear the text currently displayed in the console window. |
| SQAConsoleWrite | Write the specified text to the console window. |
| SQAEnvCreateBaseline | Capture a snapshot of the environment state before one or more tasks are performed that change or are suspected of changing the environment. |
| SQAEnvCreateCurrent | Capture a snapshot of the environment state just after some task is performed that changes or is suspected of changing the environment. |
| SQAEnvCreateDelta | Create a comparison report of the data captured in the pre-task and post-task snapshots. |
| SQAGetCaptionTerminatorChar | Retrieve the character that Robot is currently using as the window caption terminator character. |
| SQAGetDir | Retrieve the path of standard directories used by Rational test applications. |
| SQAGetLogDir | Retrieve the path of the runtime log. |
| SQAGetOcrRegionRect | Retrieve the coordinates of the specified OCR region. |
| SQAGetOcrRegionText | Retrieve the text in the specified OCR region. |
| SQAGetSystemLong | Retrieve a system value. |
| SQALogMessage | Write a message to a log and optionally insert a result flag (Pass, Fail, or Warning) in the Result column. |
| SQAQueryKey | Return the state of a locking key (Caps Lock, Num Lock, and Scroll Lock). |
| SQAResumeLogOutput | Resume the output of verification point and wait state results to the log. |
| SQAScriptCmdFailure | Generate a script command failure. |
| SQASetAssignmentChar | Set the character to be used by Robot as the assignment character in SQABasic statements. |

Utility Commands (SQABasic Additions)

| | |
|---------------------------------|---|
| SQASetCaptionTerminatorChar | Set the character that Robot uses as the window caption terminator character. |
| SQASetDefaultBrowser | Set the default browser to use during playback. |
| SQASetSeparatorChar | Set the character to be used by Robot as the separator character in SQABasic statements. |
| SQAShellExecute | Open an application or a file. |
| SQASuspendLogOutput | Suspend the output of verification point and wait state results to the log. |
| SQAVpGetActualFileName | Generate a unique path and name for an actual data file used in a custom verification point. |
| SQAVpGetBaselineFileName | Generate a unique path and name for a baseline data file used in a custom verification point. |
| SQAVpGetCurrentBaselineFileName | Generate the path and name for the current baseline data file used in a custom verification point. |
| SQAVpLog | Write a custom verification point record to a log. |
| StartApplication | Start the specified application from within the currently running script. |
| StartBrowser | Start an instance of a Web browser. |
| StartJavaApplication | Start the specified Java application from within the currently running script. |
| StartSaveWindowPositions | Mark the start of the script commands that save the window positions for restoration at playback. |
| StartTimer | Start the specified timer in the currently running script and write a message to the log. |
| StopTimer | Stop the specified timer in the currently running script and write the elapsed time in milliseconds to the log. |

NOTE: The command names now prefixed by SQA were prefixed by PLA in previous releases. The old form of each name should no longer be used, but it continues to be supported to maintain the upward compatibility of your existing scripts.

NOTE: WriteLogMessage has been replaced by SQALogMessage.

Variants

| | |
|---------|--|
| IsEmpty | Determine whether a variant has been initialized. |
| IsNull | Determine whether a variant contains a NULL value. |
| Null | Return a null variant. |
| VarType | Return the type of data stored in a variant. |

Verification Point Commands (SQABasic Additions)

These commands compare the results of a user action captured during *playback* against the result of the same action captured during *recording*. If the playback result matches the recorded baseline, the verification point passes. If the result is different, the verification point fails:

| | |
|------------------|---|
| AnimateControlVP | Establish a verification point for an animation control. |
| CalendarVP | Establish a verification point for a month calendar control. |
| CheckBoxVP | Establish a verification point for a check box control. |
| ClipboardVP | Establish an alphanumeric verification point for the contents of the Windows Clipboard. |
| ComboBoxVP | Establish a verification point for a combo box control. |
| ComboEditBoxVP | Establish a verification point for a combo edit box control. |
| ComboListBoxVP | Establish a verification point for a combo list box control. |
| DataWindowVP | Establish a verification point for a PowerBuilder DataWindow. |
| DateTimeVP | Establish a verification point for a date and time picker (DTP) control. |
| EditBoxVP | Establish a verification point for an edit box control. |
| FileVP | Establish a verification point for a file or files. |
| GenericObjectVP | Establish a verification point for a generic object. |
| GroupBoxVP | Establish a verification point for a group box control. |
| HeaderVP | Establish a verification point for a header control. |
| HotKeyControlVP | Establish a verification point for a hot key control. |
| HTMLVP | Establish a verification point for an HTML tag. |
| HTMLActiveXVP | Establish a verification point for an ActiveX control embedded in the page. |

Verification Point Commands (SQABasic Additions)

| | |
|-------------------|--|
| HTMLDocumentVP | Establish a verification point for Web page data. |
| HTMLHiddenVP | Establish a verification point for a hidden element. |
| HTMLImageVP | Establish a verification point for a Web page image. |
| HTMLLinkVP | Establish a verification point for a Web page link. |
| HTMLTableVP | Establish a verification point for a Web page table. |
| IPAddressVP | Establish a verification point for an IP Address control. |
| JavaCanvasVP | Establish a verification point for a Java canvas component. |
| JavaListViewVP | Establish a verification point for a Java multi-column list component. |
| JavaMenuVP | Establish a verification point for a Java menu. |
| JavaObjectVP | Establish a verification point for an unrecognized Java component. |
| JavaPanelVP | Establish a verification point for a Java panel or canvas. |
| JavaPopupMenuVP | Establish a verification point for a Java popup menu. |
| JavaSplitPaneVP | Establish a verification point for a Java split pane. |
| JavaSplitterVP | Establish a verification point for a Java splitter. |
| JavaTableVP | Establish a verification point for a Java table. |
| JavaTableHeaderVP | Establish a verification point for a Java table header. |
| JavaTreeVP | Establish a verification point for a Java tree component. |
| JavaWindowVP | Establish a verification point for a Java window. |
| LabelVP | Establish a verification point for a label control. |
| ListBoxVP | Establish a verification point for a list box control. |
| ListViewVP | Establish a verification point for a list view control. |
| ModuleVP | Verify whether a specified module is in memory during playback. |
| PagerVP | Establish a verification point for a pager control. |
| ProgressBarVP | Establish a verification point for a progress bar control. |
| PSGridHeaderVP | Establish a verification point for a column header in a PeopleTools grid. |
| PSGridVP | Establish a verification point for a PeopleTools grid. |
| PSMenuVP | Establish a verification point for a PeopleTools menu object. |
| PSNavigatorVP | Establish a verification point for a PeopleTools Navigator window or a navigator map in the PeopleTools Business Process Designer. |
| PSPanelVP | Establish a verification point for a PeopleTools panel. |
| PSSpinVP | Establish a verification point for a PeopleTools spin control. |

Verification Point Commands (SQABasic Additions)

| | |
|----------------|---|
| PSTreeHeaderVP | Establish a verification point for a column header in a PeopleTools tree object. |
| PSTreeVP | Establish a verification point for a PeopleTools tree object. |
| PushButtonVP | Establish a verification point for a push button control. |
| RadioButtonVP | Establish a verification point for an option button control. |
| RebarVP | Establish a verification point for a rebar control. |
| RegionVP | Establish a verification point for a specified rectangular screen region. |
| RichEditVP | Establish a verification point for a rich edit control. |
| ScrollBarVP | Establish a verification point for a scroll bar control. |
| SpinControlVP | Establish a verification point for a spin control. |
| StatusBarVP | Establish a verification point for a status bar control. |
| TabControlVP | Establish a verification point for a tab control. |
| ToolBarVP | Establish a verification point for a toolbar control. |
| TrackbarVP | Establish a verification point for a trackbar control. |
| TreeViewVP | Establish a verification point for a tree view control. |
| WebSiteVP | Test for defects (such as missing or broken links) on a Web site, or compare Web sites. |
| WindowVP | Establish a verification point for a window. |

Verification Point Commands (SQABasic Additions)

▶ ▶ ▶ Part II

Using SQABasic

▶ ▶ ▶ C H A P T E R 3

SQABasic Fundamentals

This chapter describes the following SQABasic language elements:

- ▶ Commands
- ▶ Arguments
- ▶ Data types
- ▶ Arrays
- ▶ Dynamic arrays
- ▶ Expressions and operators
- ▶ Scope of variables and constants
- ▶ Two-digit year conversions
- ▶ Trappable errors

See Appendix A for a summary of SQABasic syntax conventions.

Commands

These are the major categories of SQABasic commands:

| Command | Description | User-definable? |
|---|---|---|
| Statement | <p>A keyword that specifies an action, declaration, or definition. Examples:</p> <pre>Option Explicit GoTo ErrorRoutine Dim i As Integer Let i = 10</pre> | <i>No.</i> Statement keywords are predefined elements of the SQABasic language. |
| Function procedure (referred to as <i>functions</i>) | <p>One or more lines of code that perform a specific task. Functions return a value. Examples:</p> <pre>i = Len(MyString) RetVal = MyFunction(x) Call MyFunction(x)</pre> | <i>Yes.</i> Functions begin with the statement <code>Function</code> and end with the statement <code>End Function</code> . |
| Sub procedure | <p>One or more lines of code that perform a specific task. Sub procedures don't return values. Examples:</p> <pre>MySubProc x Call MySubProc(x)</pre> | <i>Yes.</i> Sub procedures begin with the statement <code>Sub</code> and end with the statement <code>End Sub</code> . |

See Chapter 6 for a description of the `Function...End Function` statement and the `Sub...End Sub` statement.

NOTE: A script contains one or more sub procedures. When you record a script, SQA Robot declares the sub procedure it generates as `Sub Main`.

Arguments

Most SQABasic functions and sub procedures take one or more arguments:

- ▶ If a function takes arguments, enclose the arguments in parentheses and separate them with commas.
- ▶ If a sub procedure takes arguments, separate the arguments with commas, but do not enclose the arguments in parentheses.

NOTE: If you use the Call statement to call a sub procedure, you enclose the arguments in parentheses just as you would for a function.

Passing Arguments By Value or By Reference

You can pass an argument to a function or sub procedure in one of two ways:

By value – The value of the argument variable is unchanged when the function or sub procedure returns control to the caller.

By reference – The value of the variable *can be changed* by the function or sub procedure. If the value changes, the calling function or sub procedure uses the new value in subsequent processing.

By default, values are passed by reference.

Syntax of By-Value and By-Reference Arguments

- ▶ To pass an argument **by value**, enclose the argument in parentheses. When you do this, an argument for a function (or a sub procedure called with the Call statement) is enclosed in double parentheses.
- ▶ In the following examples, the argument *x* is passed by value. The argument *y* is passed by reference:

```
Call MySub((x))
Call MySub ((x),y)
MySub(x)
MySub(x),y
z=MyFunction(x)
Call MyFunction((x))
```

- ▶ To pass an argument **by reference**, no special syntax is required.
- ▶ In the following examples, all arguments are passed by reference:

```
Call MySub(x)
Call MySub (x,y)
MySub x,y
Z=MyFunction(x)
Call MyFunction(x)
```

Syntax for Passing Arguments to External Procedures

To use a procedure stored in an external module or .DLL file, you must first `Declare` the module or procedure. The `Declare` statement uses different syntax for specifying whether arguments are to be passed by value or by reference, as follows:

- ▶ To pass an argument **by value**, use the `ByVal` statement.
- ▶ To pass an argument **by reference**, no special syntax is required. Passing an argument by reference is the default.
- ▶ For example:

```
Declare Sub MySub Lib "MyDll" (ByVal x As Integer, y As String)
```

Passing Named Arguments

When you call an SQABasic command that takes arguments, you usually supply values for those arguments by listing them in a particular order — the order in which the arguments appear in the syntax definition. This rule applies to built-in SQABasic commands as well as functions and sub procedures you create.

For example, suppose you declare a function this way:

```
Function MyFunction(id, action, value)
```

From the above syntax, you know that `MyFunction` requires three arguments: *id*, *action*, and *value*. When you call this function, you supply the arguments in the order shown in the declaration.

If a command contains just a few arguments, it's fairly easy to remember the order of the arguments. However, if a command has several arguments, and you want to be sure the values you supply are assigned to the correct arguments, consider using named arguments.

Named arguments are arguments identified by name rather than by syntax position. With named arguments, the order of the arguments is not important.

All SQABasic commands accept named arguments.

Syntax of Named Arguments

Named arguments have this syntax:

```
namedarg:= value
```

In the `MyFunction` example, both function calls below assign the correct values to the appropriate arguments:

```
MyFunction id:=1, action:="get", value:=0
MyFunction action:="get", value:=0, id:=1
```

If an argument is optional and you don't want to provide a value for the optional argument, simply omit it.

For example, if the *action* argument of the `MyFunction` call is optional, you could call the function like this:

```
MyFunction action:="get",id:=1
```

NOTE: Although you can shift the order of named arguments, you can't omit required arguments.

Data Types

You declare the data type of a variable in any of these ways:

Explicit declaration – Data types are explicitly declared with the `Dim` statement.

Type-declaration character – When first referencing a variable, you can declare the variable by adding a type-declaration character (such as `$` for `String` or `%` for `Integer`) to the end of the variable name.

Implicit declaration – If neither a `Dim` statement nor a type-declaration character is used to declare a variable, SQA automatically assigns the variable the default data type `Variant`.

Once a data type is declared, a variable can only contain data of the declared type.

NOTE: You must always explicitly declare variables of a `User-Defined` data type. If you use the `Option Explicit` statement, you must explicitly declare all variables.

Data Types

Descriptions of SQABasic Data Types

These are the data types SQABasic supports:

| Data type | Type character | Storage size | Range |
|--|----------------|---|--|
| Integer (short) | % | 2 bytes | -32,768 to 32,767 |
| Long (long) | & | 4 bytes | -2,147,483,648 to 2,147,483,647 |
| Single (single-precision floating point) | ! | 4 bytes | -3.402E38 to -1.401E-45 (for negative values) 1.401E-45 to 3.402E38 (for positive values) |
| Double (double-precision floating-point) | # | 8 bytes | -1.797E308 to -4.94E-324 (for negative values) 4.94E-324 to 1.797E308 (for positive values) |
| Currency | @ | 8 bytes (fixed) | -922,337,203,685,477.5808 to 922,337,203,685,477.5807 |
| String (variable length) | \$ | 0 to about 32 KB | 0 characters to 32,767 characters |
| String (fixed length) | None | 1 to about 32 KB | 1 character to 32,767 characters |
| Object | None | n/a | n/a |
| Variant | None | A Variant's storage size and range depend on the way the Variant is used. For example, a Variant used as an Integer is stored in 2 bytes and has a range between -32,768 and 32,767 | |
| User-Defined | None | Byte size is set by individual elements | The range of each element is determined by the element's declared data type |

Data Type Notes

- ▶ Variants support most of the data type in the table. The unsupported data types are fixed-length Strings and User-Defined data types.
- ▶ Variants can also be used as a Date data type. A Variant used as a date is stored as an 8-byte Double. Values range from Jan 1st, 100 to Dec 31st, 9999.
- ▶ Numeric values are always signed.
- ▶ SQABasic has no true Boolean variables. SQABasic considers 0 to be FALSE and any other numeric value to be TRUE. Only numeric values can be used as Booleans. Comparison operator expressions always return 0 for FALSE and -1 for TRUE.
- ▶ Integer constants can be expressed in decimal, octal, or hexadecimal notation. Decimal constants are expressed by simply using the decimal representation. To represent an octal value, precede the constant with &O or &o (for example, &o177). To represent a hexadecimal value, precede the constant with &H or &h (for example, &H8001).
- ▶ There are no restrictions on the characters you can include in a string. For example, the character whose ANSI value is 0 can be embedded in a string.
- ▶ See the following sections for more information about Variant and User-Defined data types.

Variant Data Types

You declare a Variant data type in either of these ways:

- ▶ Explicitly through the Dim statement.
- ▶ Implicitly by using a variable without declaring it explicitly or through a type-declaration character. By default, SQABasic assigns the data type Variant to any undeclared variable.

Valid Variant Data Types

A Variant data type can be used to store any type of data except fixed-length String data and User-Defined data.

In addition, there are these special Variant data types:

Empty Variants – Any newly-defined Variant defaults to the Variant type Empty. Empty Variants contain no initialized data.

Data Types

An `Empty Variant` is zero when used in a numeric expression, and it is an empty string when used in a string expression. Call the `IsEmpty` function to test whether a `Variant` is uninitialized (empty).

`Null Variants` – These `Variants` have no associated data and serve only to represent invalid or ambiguous results. Call the `IsNull` function to test whether a `Variant` contains a null value.

`Date Variants` – Date values range from Jan 1st, 100 to Dec 31st, 9999. See the `Format` function in Chapter 6 for information about valid date formats.

Identifying the Type of Data Stored in a Variant

A tag stored with `Variant` data identifies the type of data the `Variant` contains. You can examine the tag by calling the `VarType` function.

User-Defined Data Types

A `User-Defined data type` is a set of related variables that can be referenced by a single variable name. It is similar to a C data structure.

`User-Defined data types` contain one or more elements. An **element** in a `User-Defined data type` can contain any type of data that `SQABasic` supports. An element can also contain an array or another `User-Defined type`.

Declaring a Variable as a User-Defined Data Type

Before you can declare a variable as a `User-Defined data type`, you first must define the data type. You can then declare as many variables of that type as you like — just as you can declare as many variables as you like of type `Integer` or `String`.

Here are the basic steps for defining a `User-Defined type`:

1. Use the `Type` statement to define the `User-Defined data type`, as in:

```
Type CustData                                ' Name of the data type
    CustName As String                        ' Element for customer's name
    CustID As Long                            ' Element for customer's ID
End Type
```

2. Use the `Dim` statement to declare a variable of the type you just defined:

```
Dim Customer As CustData                    ' Declare the variable Customer
```

Use dot-notation syntax to reference an individual element — for example:

```
Customer.CustName = "Jennifer Farriday"
Customer.CustID = 533128
```

Dialog Box Records

In SQABasic, you create a dialog box by first defining a dialog box record. **Dialog box records** look like any other user-defined data type, but there are two important differences:

- ▶ You define a dialog box record with the `Begin Dialog . . . End Dialog` statements, not the `Type . . . End Type` statements.
- ▶ The elements in a dialog box record refer to the objects (such as buttons, entry fields, and labels) in the dialog box.

Once you define a dialog box record, you declare an instance of that record. Like other user-defined types, you use the `Dim` statement to declare an instance of a dialog box. Also, you use dot-notation syntax to refer to the objects in a dialog box:

```
MyDialog.Columns = "2"
```

See the `Begin Dialog` statement in Chapter 6 for more information about creating dialog boxes.

Data Type Conversions

SQABasic attempts to convert one dissimilar data type to another when moving data between the following data types:

- ▶ Between any two numeric types – When converting from a larger type to a smaller type (for example, a `Long` to an `Integer`), a runtime numeric overflow error might occur. This error indicates that the number of the larger type is too large for the target data type. For example, loss of precision is not a runtime error when converting from `Double` to `Single`, or from either float type to either `Integer` type.
- ▶ Between fixed-length strings and dynamic (variable-length) strings – When converting a fixed-length string to dynamic, a dynamic string that has the same length and contents as the fixed-length string is created. When converting from a dynamic string to a fixed-length string, some adjustment might be required. If the dynamic string is shorter than the fixed-length string, the resulting fixed-length string is extended with spaces. If the dynamic string is longer than the fixed-length string, the resulting fixed-length string is a truncated version of the dynamic string. No runtime errors are caused by string conversions.
- ▶ Between any data type and **variant** data types – Any data type (other than a `User-Defined` type) can be converted to a `Variant` data type. SQABasic converts variant strings to numbers when required. A type mismatch error occurs if the variant string does not contain a valid representation of a number.

Arrays

No other implicit conversions are supported. In particular, SQABasic does not automatically convert between numeric and string data. Use the functions `Val` and `Str$` for such conversions.

Arrays

An **array** is a variable made up of individual elements that have the same data type. Each element is accessed through a unique index number.

An array has one or more **dimensions** (sets of elements). An array can have up to 60 dimensions.

Array **subscripts** specify the number of elements in a dimension by setting its starting and ending index values. For example, the following array `MyArray` has one dimension with a starting index value of 1 and an ending index value of 100:

```
Dim MyArray(1 To 100) As String
```

If only one subscript is provided (which is typically the case), it is assumed to specify the ending index value. The starting index value defaults to 0. You can set the starting index default to either 0 or 1 through the `Option Base` statement.

Arrays support all SQABasic data types. Arrays of arrays and dialog box records are not supported.

Declaring an Array

The following array has two dimensions containing 11 elements and 101 elements, respectively (the default starting index is 0 for each dimension):

```
Dim MyArray (10,100) as Integer
```

See the `Dim` statement in Chapter 6 for more information.

Referencing an Array

You reference array elements by enclosing the proper index values in parentheses after the array name – for example, `ArrayName(i, j) = x`.

Dynamic Arrays

When you declare a **dynamic array**, you don't specify a subscript range for the array elements. Instead, you use the `ReDim` statement to set the subscript range.

The advantage of using dynamic arrays is that you can base the number of array elements on unpredictable conditions that only become known at runtime. Because you don't have to pre-define the number of elements in the array, you avoid having to reserve space for elements that you might not use.

For example, suppose you want to use an array to store a set of values entered by a user, but you don't know in advance how many values the user needs to store. In this case, you dimension the array without specifying a subscript range, and then you execute a `ReDim` statement to increase the range by 1 each time the user is about to enter a new value. Or, you might want to prompt for the number of values the user wants to enter, and then execute one `ReDim` statement to set the size of the array accordingly before prompting for the entry.

NOTE: `ReDim` destroys the current contents of the array. To preserve the array's contents, include the `Preserve` argument in your `ReDim` statement.

Dimensions of a Dynamic Array

If you `Dim` a dynamic array before using it, the maximum number of dimensions it can have is 8. To create dynamic arrays with more dimensions (up to 60), do not `Dim` the array at all. Instead, use the `ReDim` statement inside your procedure.

Dynamic Array Example

In this example, the dynamic array `varray` contains user-defined cash flow values:

```
Sub main
  Dim aprate as Single
  Dim varray() as Double
  Dim cflowper as Integer
  Dim msgtext
  Dim x as Integer
  Dim netpv as Double
  cflowper=InputBox("Enter number of cash flow periods")
  ReDim varray(cflowper)
  For x= 1 to cflowper
    varray(x)=InputBox("Enter cash flow for period #" & x & ":")
  Next x
  aprate=InputBox("Enter discount rate: ")
  If aprate>1 then
    aprate=aprate/100
  End If
  netpv=NPV(aprate,varray())
  msgtext="The net present value is: "
  msgtext=msgtext & Format(netpv, "Currency")
  MsgBox msgtext
End Sub
```

Expressions and Operators

An **expression** is a collection of two or more terms that perform a mathematical, comparative, or logical operation. The type of operation performed is determined by the **operator** in the expression.

Expressions are evaluated according to an established order of precedence for operators. Use parentheses to override the default precedence order.

Operator precedence order (from high to low) is:

- Numeric operators
- String concatenation operators
- Comparison operators
- Logical operators

Numeric Operators

Numeric operators are shown in order of precedence (from high to low):

| Operator | Description |
|---------------------------------|--|
| <code>^</code> | Exponentiation. |
| <code>- , +</code> | Unary minus and plus. |
| <code>*</code> , <code>/</code> | Numeric multiplication or division. For division, the result is a <code>Double</code> . |
| <code>\</code> | Integer division. The operands can be <code>Integer</code> or <code>Long</code> . |
| <code>Mod</code> | Modulus or Remainder. The operands can be <code>Integer</code> or <code>Long</code> . |
| <code>- , +</code> | Numeric addition and subtraction. The <code>+</code> operator can also be used for string concatenation. |

String Concatenation Operators

The string concatenation operator is the ampersand (`&`). Alternatively, you can use a plus sign (`+`).

Comparison Operators

Comparison operators have equal precedence. They are evaluated from left to right:

| Operator | Description |
|----------|--------------------------|
| > | Greater than |
| < | Less than |
| = | Equal to |
| <= | Less than or equal to |
| >= | Greater than or equal to |
| <> | Not equal to |

Comparison operators compare numbers and strings:

- ▶ For numbers, operands are widened to the least common type (`Integer` is preferred over `Long`, `Long` is preferred over `Single`, and `Single` is preferred over `Double`).
- ▶ For English strings, comparisons are case-sensitive by default. You can change the default through the `Option Compare` statement.
- ▶ String comparisons return 0 for FALSE and -1 for TRUE.

Logical Operators

Logical operators are shown in order of precedence (from high to low):

| Operator | Description |
|----------|---|
| Not | Unary Not – operand can be <code>Integer</code> or <code>Long</code> . The operation is performed bitwise (one's complement). |
| And | And – operands can be <code>Integer</code> or <code>Long</code> . The operation is performed bitwise. |
| Or | Inclusive Or – operands can be <code>Integer</code> or <code>Long</code> . The operation is performed bitwise. |

▶ ▶ ▶

Scope of Variables and Constants

▶ ▶ ▶

| Operator | Description |
|----------|--|
| Xor | Exclusive Or – operands can be Integer or Long. The operation is performed bitwise. |
| Eqv | Equivalence – operands can be Integer or Long. The operation is performed bitwise. (A Eqv B) is the same as (Not (A Xor B)). |
| Imp | Implication – operands can be Integer or Long. The operation is performed bitwise. (A Imp B) is the same as ((Not A) Or B). |

Scope of Variables and Constants

The scope of variables and constants can be any of the following:

- ▶ Local. Accessible only to the function or sub procedure containing the variable or constant declaration. Use the statement `Dim` to declare local variables and `Const` for local constants.
- ▶ Module-level. Accessible to any function or sub procedure in the same module (script or library file) as the `Dim` or `Const` statement. With module-level declarations, place the `Dim` or `Const` statement above the first procedure in the module.
- ▶ Global. Accessible to any function or sub procedure in any module. Use the `Global` statement for global declarations. Global declarations can appear in a module or in a header file.

For more information about the scope of variables and constants, including how to declare each type, see the section *Declaring Variables and Constants* in Chapter 4, *SQABasic Scripts*.

For information on module-level and global procedures, see the sections *Adding Custom Procedures to a Script* and *Adding Custom Procedures to a Library File* in Chapter 4, *SQABasic Scripts*.

Year 2000 Compliance

SQABasic converts two-digit years to four-digit years in the following situations:

| Command or assignment | Two-digit year conversion |
|---|--|
| Input string with a two-digit year, when converted to an internal date value | 00 through 29 is converted to 2000 through 2029 30 through 99 is converted to 1930 through 1999 |
| Date statement | 80 through 99 is converted to 1980 through 1999 00 through 79 is converted to 2000 through 2079 |
| Format function with the following <i>format</i> argument values: General Date Short Date c d | Two-digit dates are formatted as four-digit dates |
| CVDate, DateValue, and Year functions | 00 through 29 is converted to 2000 through 2029 30 through 99 is converted to 1930 through 1999 |

Of course, you can force a two-digit year through a user-defined date format — for example:

```
Sub Main
  Dim datestr

  datestr = InputBox("Enter a date with a 2-digit year" + _
    Chr$(13) + "(in the format mm/dd/yy):")

  'CVDate converts to a 4-digit year
  datestr = CVDate(datestr)
  MsgBox "Default format: " + datestr

  'Now change the format to use a 2-digit year
  datestr = Format(datestr, "m/d/yy")
  MsgBox "Custom format: " + datestr
End Sub
```

Suggestions for Avoiding Year 2000 Problems

Here are some guidelines for avoiding year 2000 problems in your scripts:

- ▶ Always maintain internal date information as date values.
- ▶ Store date values in variables with numeric or variant data types.
- ▶ Use date values, not strings, when performing date calculations.
- ▶ When accepting date information from the user, always display the value received in a format that explicitly identifies the century.
- ▶ When displaying data information, always use a format that explicitly identifies the century.
- ▶ When exchanging data information with external data sources or external programs, you should use double-precision floating point numbers or data strings with at least four characters for identifying the century.

Trappable Errors

Trappable errors are runtime errors that you can respond to in any way you choose. If you don't provide a response to a trappable error, SQABasic displays an error dialog box at runtime.

SQABasic provides the following error-handling commands:

| Command | Description |
|---------------------------------|---|
| <code>Err</code> statement | Sets a runtime error code without simulating an occurrence of the error |
| <code>Err</code> function | Returns the error code for the last error trapped |
| <code>Error</code> statement | Simulates the occurrence of a runtime error |
| <code>Error</code> function | Returns the error message that corresponds to the specified error code |
| <code>On Error</code> statement | Specifies how your program responds to a runtime error |

Error codes aren't automatically returned. You must retrieve them with `Err`.

See Appendix B for a list of trappable error codes.

Responding to Errors

You can respond to errors in either of these ways:

- ▶ Put error-handling code directly before a line of code where an error might occur (such as after a `File Open` statement).
- ▶ Create a separate section of the procedure just for error handling, and assign the section an appropriate label. When an error occurs, program flow jumps to the label.
- ▶ You typically use this method to test for and react to different error codes.

Use the `On Error` statement to specify either method.

User-Defined Errors

In addition to the standard runtime errors reported by SQABasic, you might want to create your own set of codes for trapping errors specific to your program. For example, if your program establishes rules for file input, you might want to trap for errors that result when the user doesn't follow the rules.

You can trigger an error and respond appropriately through the same statements and functions you use for standard SQABasic error codes.

Error-Handling Examples

SQABasic online Help contains examples of how you can respond to runtime errors. To see the examples:

1. Choose Using SQABasic from Contents.
2. Choose Error Handling.
3. Choose Trapping Errors Returned by SQABasic or Trapping User-Defined (Non-SQABasic) Errors.

Trappable Errors

▶ ▶ ▶ C H A P T E R 4

SQABasic Scripts

Rational Robot automatically generates scripts for you during recording. However, because you may want to edit the scripts that Robot generates, and even create custom procedures and library files, you should have a fundamental understanding of the structure and contents of a script.

This chapter includes the following topics:

- ▶ What is a script?
- ▶ User action and verification point commands
- ▶ Object context
- ▶ Customizing scripts

What is a Script?

A **script** is an ASCII text file that contains SQABasic commands. A compiled script can be executed (played back) by Robot or by the `CallScript` command.

When you record a script, Robot translates your actions into a series of SQABasic commands and stores them in the script. When you play back the script, Robot performs the actions you recorded by executing the SQABasic commands.

Typically, GUI scripts include user actions such as mouse clicks and keystrokes. GUI scripts also include verification points that you insert during recording.

Scripts that Robot generates consist of a single sub procedure called `Main`. Optionally, you can add custom sub procedures and functions to the script file, as described in the section *Adding Custom Procedures to a Script* on page 4-23.

NOTE: A script is also associated with properties such as the purpose of the script and the type of script. Typically, you define script properties when you plan the script with TestManager. You can also view and edit script properties in Robot.

What is a Script?

Script Source Files

GUI scripts have the extension `.rec`.



If changes are made to a script, Robot automatically saves the script when you compile it, play it back, or debug it. To explicitly save a script during editing, click `File → Save`, or click the `Save` button on the toolbar.

Script Executable Files

A compiled script has the extension `.sbx`. Only Robot can execute a `.sbx` file.



At the start of playback or debugging, Robot automatically compiles a script if it has changed since it last ran. To explicitly compile a script during editing, click `File → Compile`, or click the `Compile` button on the toolbar.

Script Structure

The typical `Main` sub procedure that Robot generates in a script can be broken into four general sections:

- ▶ Initialization
- ▶ Window restoration (optional)
- ▶ Script body (window context, user actions, and verification points)
- ▶ Close

Script Initialization

All Robot scripts begin with the following commands:

- ▶ `Sub Main`

Defines a subroutine named `Main`. This is normally the first command in the script and should not be edited.

The name `Main` is reserved for scripts Robot generates. Do not assign this name to any custom procedures you may write.
- ▶ `Dim Result As Integer`

Defines the variable `Result` as an integer variable. Robot returns values from verification point commands into the variable `Result`. The value of `Result` is local to the `Main` subroutine.


```
► 'Initially Recorded: 06/16/98 14:08:33
   'Script Name: CdOrder
```

Robot writes two comment lines in the initialization section of each script. The first shows when the script was recorded, and the second shows the script name. These comments are not required and can be edited or removed.

Window Restoration

Robot includes the following two commands at the beginning of a script if Save window positions is selected in the General tab of the GUI Record Options dialog box:

```
StartSaveWindowPositions           ' Window restoration commands
. . .
EndSaveWindowPositions
```

During playback, the window restoration commands bracketed between `StartSaveWindowPositions` and `EndSaveWindowPositions` restore the specified windows to the size and position they were in at the start of recording. Also, a context window (a window within which subsequent user actions are to occur) may be specified — for example, with MDI applications.

The referenced windows must exist during playback before the window restoration commands can be properly executed.

`StartSaveWindowPositions` and `EndSaveWindowPositions` also tell Robot that, during playback, the intervening `Window SetContext`, `Window MoveTo`, and `Window SetPosition` commands are for window restoration only. During window restoration, all playback timing defaults are set to zero in order to process the commands as quickly as possible.

If any command fails between `StartSaveWindowPositions` and `EndSaveWindowPositions`, that failure is reported to the log as a warning, not as a script command failure.

NOTE: Additionally, you can save the positions of all active windows (except hidden windows) after every `Window SetContext` command by selecting the GUI recording option Auto Record Window Size (on the General tab). During playback, Robot restores the windows to their positions when the script was recorded. Robot writes warning messages to the log for any windows it can't find during playback.

What is a Script?

Script Body

The script body is the primary processing section of the script. The script body typically includes SQABasic commands that:

- ▶ Perform *user actions* — for example, keystrokes and mouse clicks you make to navigate through the application and to provide data to the application.

For more information, see *User Action Commands* on page 4-6.

- ▶ Establish *verification points* by comparing information captured for an object during recording with information captured for the object during playback.

For more information, see *Verification Point Commands* on page 4-7.

- ▶ Set the *context window*. When you set the context window, Robot expects subsequent actions and verification points to be performed within that window.

For more information, see *Establishing Context through a Window Command* on page 4-15.

Script Close

All scripts that Robot generates end with the following command. This command terminates the script.

```
End Sub
```

This line indicates the end of the `Main` subroutine.

Sample Script

The following short script illustrates the four sections of a script as well as typical actions you can record in a script.

In this example, the application-under-test is `Classics.exe`, a Visual Basic application for ordering CDs. As the user places an order for two CDs of the same title, Robot records the user's actions. In the dialog box where the user provides credit card and other ordering information, the user performs verification points on the following dialog box objects:

- ▶ `txtAlbumInfo` – An edit box that displays the name of the CD being purchased.
- ▶ `txtQuantity` – An edit box that displays the number of CDs ordered.
- ▶ `lblTotal` – A non-modifiable label object that displays the cost of the order.

| | |
|---|--|
| <p><i>Initialization</i></p> | <pre>Sub Main Dim Result As Integer 'Initially Recorded: 06/16/98 16:09:16 'Script Name: CdOrder</pre> |
| <p><i>Window Restoration (optional)</i></p> | <pre>' Restore all windows to their size and position during recording StartSaveWindowPositions Window SetPosition, "Caption=Program Manager", "Coords=0,0,1024,768;Status=NORMAL" Window SetPosition, "Caption=Exploring - C:\Classics\AccessData", "Coords=-32000,-32000,160,24;Status=MINIMIZED" Window SetPosition, "Caption=Untitled - Notepad", "Coords=76,18,558,418;Status=NORMAL" Window SetPosition, "Caption=Microsoft Excel - Book1", "Coords=363,247,639,460;Status=NORMAL" Window SetContext, "Caption=Microsoft Excel - Book1", "" Window SetPosition, "Caption=Book1;ChildWindow", "Coords=-6,-25,639,349;Status=NORMAL" Window SetPosition, "Class=Shell_TrayWnd", "Coords=-2,740,1028,30;Status=NORMAL" EndSaveWindowPositions</pre> |
| <p><i>Script body</i></p> <ul style="list-style-type: none"> ▪ Context window ▪ User actions ▪ Verification points | <pre>' Start the application-under-test StartApplication "C:\Classics Online\Classics.exe" ' Select the title of the CD to purchase Window SetContext, "Name=frmMain", "" TreeView Click, "Name=treMain;\;ItemText=Bach->Brandenburg Concertos Nos. 1 3", "" PushButton Click, "Name=cmdOrder" ' Login Window SetContext, "Name=frmOrderLogin", "" PushButton Click, "Name=cmdOR" ' Specify the number of CDs to purchase Window SetContext, "Name=frmOrder", "" EditBox Left_Drag, "Name=txtQuantity", "Coords=25,10,-120,11" InputKeys "2" ' Provide credit card information ComboBox Click, "Name=comboCardType", "Coords=104,7" ComboBox Click, "ObjectIndex=1", "Text=MasterCard" EditBox Click, "Name=txtCreditCard", "Coords=49,11" InputKeys "1535399178421813" EditBox Click, "Name=txtExpirationDate", "Coords=11,5" InputKeys "12/31/00" ' Verify that the correct CD is being purchased Result = EditBoxVP (CompareText, "Name=txtAlbumInfo", "VP=TitleText;Type=CaseSensitive")</pre> |

User Action and Verification Point Commands

| | |
|---|---|
| <p><i>Script body (Cont.)</i></p> <ul style="list-style-type: none">▪ Context window▪ User actions▪ Verification points | <pre>' Verify that the number of CDs being purchased is correct Result = EditBoxVP (CompareText, "Name=txtQuantity", "VP=QuantityText;Type=CaseInsensitive") ' Verify the correct total purchas price Result = LabelVP (CompareProperties, "Name=lblTotal", "VP=CostObjProp") ' Close the application-under-test PushButton Click, "Name=cmdCancel" Window SetContext, "Name=frmMain", "" Window CloseWin, "", ""</pre> |
| <p>Close</p> | <pre>End Sub</pre> |

User Action and Verification Point Commands

To read or edit a script successfully, you need to have a basic understanding of two important categories of commands that are executed within the body of a script. These categories are:

- ▶ User action commands
- ▶ Verification point commands

The following sections describe these commands.

User Action Commands

User actions include all of the GUI actions you perform during recording — for example, clicking a button that opens a dialog box, selecting an item in a list, or typing data into an order form.

You perform user actions as you navigate through the application-under-test and as you supply data to the application-under-test.

User action command names (such as `PushButton`, `Window`, or `EditBox`) reflect the object being acted upon. User action command names are followed by the *action* argument (containing values such as `Click`, `Resize`, or `VScrollTo`), which specifies the action taken against the object — for example:

```
PushButton Click, "Name=cmdOK"
```

For a summary of all user action commands, see the section *User Action Commands (SQABasic Additions)* in Chapter 2, *Functional List*.

Verification Point Commands

In functional testing, you need to verify that the objects in the application-under-test look and work as designed from build to build. To accomplish this, you establish **verification points** for the objects. Here is an overview of how verification points work:

- ▶ During *recording*, a verification point command captures information about an object — for example, the size, position, and other properties of the object, or any data that might be associated with the object. Information captured during recording establishes a **baseline** for future tests. The information is stored in a baseline data file and written to the LogViewer.
- ▶ During *playback*, the same verification point command again captures information about the object. The information captured during playback is compared against the baseline information captured for the object during recording — thus verifying whether the information is the same or has changed.

If there is a discrepancy between the baseline data and the data captured during playback, the latter is stored in an **actual data file** and written to the LogViewer.

At any time, you can re-record a verification point for an object, thus establishing a new baseline. For example, if the position of a push button changes in build 20 of the application-under-test, you need to record a new baseline for the push button to verify its new position in subsequent builds.

Verification point command names (such as `PushButtonVP`, `WindowVP`, or `EditBoxVP`) reflect the object you are verifying. Verification point command names are followed by the *action* argument (containing values such as `CompareData`, `CompareText`, or `CompareProperties`), which indicates the type of verification you are performing on the object — for example:

```
Result = EditBoxVP (CompareText, "Name=txtQuantity",
                   "VP=QuantityText;Type=CaseInsensitive")
```

User Action and Verification Point Commands

Verification point commands return a value to the *Result* variable. If the information captured during playback matches the baseline, the verification point passes, and *Result* equals 1. If there is no match, the verification point fails, and *Result* equals 0.

For a summary of all the verification point commands, see the section *Verification Point Commands (SQABasic Additions)* in Chapter 2, *Functional List*.

Syntax of User Action and Verification Point Commands

Syntax conventions for user action and verification point commands are similar.

The general format for a user action command is:

```
ObjectType action, recMethod, parameters
```

The general format for a verification point command is:

```
Result = ObjectTypeVP (action, recMethod, parameters)
```

Here is a summary of the key syntax elements:

| Syntax Element | Description |
|---------------------|--|
| <i>ObjectType</i> | The command name. User action command names always begin with the name of the object being acted upon — for example, <code>ComboBox</code> . When you record an action against an object, Robot automatically determines the object type. |
| <i>ObjectTypeVP</i> | Like user action command names, verification point command names indicate the target object. However, verification point names include the suffix <code>VP</code> — for example, <code>ComboBoxVP</code> . |
| action | The action performed against the object, or the type of verification point established for the object — for example: <code>ComboBox Click, "Name=lstUserName", "Coords=75,6"</code> |
| recMethod | Information Robot uses to identify and locate the target object during playback — for example: <code>ComboBox Click, "Name=lstUserName", "Coords=75,6"</code> Use double quotation marks to delimit the recognition method string. |

▶ ▶ ▶

▶ ▶ ▶

| Syntax Element | Description |
|----------------|---|
| | <p>If multiple recognition method values are needed to uniquely identify an object, enclose the entire recognition method string within a single set of quotes. For more information about multiple component values in a recognition method string, see <i>Components of a Recognition Method String</i> on page 4-10.</p> <p>Because multiple values might be required to uniquely identify an object, each object is associated with an <i>ordered</i> set of possible recognition method values that Robot can use to identify the object. For information about the order of recognition method values, see <i>Recognition Method Order</i> on page 4-10.</p> <p><i>recMethod</i> can have up to 2,048 characters.</p> |
| parameters | <p>Any additional information required by the <i>action</i> argument. For example, if the action is a mouse click on a combo box, <i>parameters</i> might contain the coordinates of the click relative to the combo box — as in:</p> <pre>ComboBox Click, "Name=lstUserName", "Coords=75,6"</pre> <p>Use double quotation marks to delimit parameters. If multiple parameter values are listed, enclose them all in a single set of quotes, and use semicolons to separate the individual values.</p> <p><i>parameters</i> can have up to 968 characters.</p> |
| Result | <p>A variable that specifies whether a verification point passes (value is 1) or fails (value is 0) during playback.</p> |

NOTE: The *recMethod* argument is also used in Object Scripting commands. For more information, see the section *Object Scripting* in Chapter 5, *Enhancements to Recorded Scripts*.

Components of a Recognition Method String

Robot uses the recognition method (*recMethod*) argument of user action and verification point commands to uniquely identify the target object.

Sometimes, more than one recognition method value is required to uniquely identify an object. If a recognition method string consists of multiple component values, enclose the entire string within a single set of quote marks ("").

A recognition method string can have two types of component values:

- ▶ Values that further define, or *qualify*, the object. These types of values are delimited by a semicolon (;). For example, the recognition method string in this command identifies a window titled Classics Online:

```
Window SetContext, "Caption=Classics Online;Class=#32770", ""
```

- ▶ Values that show a *hierarchy of objects*, such as a window and an object in that window. These types of values are delimited by a semicolon, backslash, and semicolon (; \ ;). For example, the recognition method string in this command identifies an item in a tree view object named treMain:

```
TreeView Click, "Name=treMain;\;ItemText=Haydn", "Location=Button"
```

In this example, the tree view object is in the current context window. You can also use context notation to specify an object. For more information, see *Establishing Context through Context Notation* on page 4-18.

Recognition Method Order

There are many possible pieces of information that Robot can use to uniquely identify an object. Choosing the right recognition method balances script reliability and readability.

Most of the standard object types are associated with a pre-defined, ordered list of recognition method values. While recording an action on an object, Robot tries each listed value in sequence until it can uniquely identify the object. In most cases, the object can be uniquely identified through the first value in the list for that object type, but occasionally additional information is required.

The following table lists the object types that Robot supports for user action and verification point commands, and also the default order of recognition method values it checks for each object type:

| Object type | | Order of recognition method values |
|-------------------|--------------------|------------------------------------|
| AnimateControl/VP | JavaTableHeader/VP | <i>ObjectName</i> |
| CheckBox/VP | JavaTree/VP | <i>Text</i> |
| DataWindow/VP | JavaWindow/VP | <i>Index</i> |
| GroupBox/VP | Label/VP | <i>ID</i> |
| Header/VP | ListView/VP | |
| HTML/VP | Pager/VP | |
| HTMLActiveX/VP | ProgressBar/VP | |
| HTMLDocument/VP | PSCalendar/VP | |
| HTMLHiddenVP | PSGridHeader/VP | |
| HTMLImage/VP | PSNavigator/VP | |
| HTMLink/VP | PSPanel/VP | |
| HTMLTable/VP | PSTreeHeader/VP | |
| JavaCanvas/VP | PushButton/VP | |
| JavaListView/VP | RadioButton/VP | |
| JavaMenu/VP | Rebar/VP | |
| JavaObject/VP | SpinControl/VP | |
| JavaPanel/VP | TabControl/VP | |
| JavaPopupMenu/VP | Toolbar/VP | |
| JavaSplitPane/VP | Trackbar/VP | |
| JavaSplitter/VP | TreeView/VP | |
| JavaTable/VP | | |
| ComboBox/VP | Listbox/VP | <i>ObjectName</i> |
| ComboEditBox/VP | PSGrid | <i>Label</i> |
| ComboListBox/VP | PSMenu/VP | <i>Index</i> |
| EditBox/VP | PSSpin/VP | <i>ID</i> |
| HotKeyControl/VP | PSTree | |
| IPAddress/VP | RichEdit/VP | |
| Calendar/VP | | <i>ObjectName</i> |
| DateTime/VP | | <i>Label</i> |
| | | <i>Text</i> |
| | | <i>Index</i> |
| | | <i>ID</i> |

▶ ▶ ▶

User Action and Verification Point Commands

▶ ▶ ▶

| Object type | Order of recognition method values |
|------------------------------|--|
| ScrollBar/VP StatusBar/VP | <i>ObjectName</i> <i>Index</i> <i>ID</i> |
| Desktop | None (the Windows desktop is automatically recognized) |
| Window/VP | <i>ObjectName</i> <i>Caption</i> <i>CaptionClass</i> * <i>Class</i> * Writes both <i>Caption</i> and <i>Class</i> to the script. |
| GenericObject/VP | <i>Object Name</i> <i>Text</i> <i>ClassIndex</i> * <i>Index</i> <i>ID</i> * Writes both <i>Class</i> and <i>ClassIndex</i> to the script. |

NOTE: In C++ development environments, the default order of recognition method values is different from the order shown in this table. See the next section for more information.

Changing the Default Order

You can view and optionally modify the order of recognition method values for a given object type. To do so:

1. In Robot, click Tools → GUI Record Options.
2. Click the Object Recognition Order tab.
3. Select an object type in the Object type box.
4. View and optionally modify the order of recognition method values in the Recognition method order box.

Robot can more efficiently identify the objects in a C++ application if you change the default recognition method order that it uses for other application environments. To change the recognition method order for all object types in C++ applications, select C++ Recognition Order in the Object Order Preference list.

Recognition Methods in Java Commands

When recording actions against Java objects, Robot is aware of a parent object and a child object. The parent object is the outermost Java container — for example, a frame with Java applications, or an applet with Java applets. The child object is the object being acted upon. Robot ignores any objects between the parent object and the target child object.

The *recMethod* argument in Java commands always specifies the child object. The parent object can be specified in either of these ways:

- ▶ Through the same *recMethod* argument that specifies the child object.

If a recognition method in a Java command specifies both the parent and child objects, the objects are separated by a semicolon, backslash, and semicolon (`;\;`), which is standard syntax for hierarchical objects in all recognition method strings. Here is an example:

```
JavaTree Expand, "Name=Main;\;Type=JavaTree;Name=Music",
                "Text=Music->Jazz"
```

- ▶ Through a preceding *Browser* command.

If a recognition method in a Java command doesn't explicitly specify the parent object, the parent object must be specified through a preceding *Browser* command. To specify a parent Java object, the *Browser* command includes the *action* *SetApplet* and an appropriate *recMethod* (*Name*, *JavaCaption*, or *JavaClass*, and possibly the qualifier *Index*).

Here is an example of a *Browser* command specifying a parent object named *Main*:

```
Browser SetApplet, "Name=Main", ""
JavaTree Expand, "Type=JavaTree;Name=Music", "Text=Music->Jazz"
```

The parent object in a *Browser SetApplet* command applies to all subsequent Java commands that do not explicitly specify a parent object in *recMethod*.

Using Object Scripting Commands with Java Objects

Object Scripting commands (such as `SQAGetChildren`, `SQAGetProperty`, and `SQAInvokeMethod`) cannot extract information about parent Java objects from a preceding Browser command, as other commands can. As a result, the *recMethod* argument of an Object Scripting command *must* include the parent object and child object, separated by a semicolon, backslash, and semicolon (`; \ ;`).

When you're editing your script, simply copy the parent object information from the *recMethod* argument of the preceding Browser command into the *recMethod* argument of the Object Scripting command.

For a list of the SQABasic Object Scripting commands, see *Object Scripting Commands (SQABasic Additions)* in Chapter 2.

Specifying Parent Objects in *recMethod*

When you record user actions or verification points against Java objects, Robot can write the following kinds of commands to the script:

- ▶ Commands used only with objects in the Java environment — for example, `JavaMenu`, `JavaPanel`, or `JavaTree`. These commands have the prefix `Java`.
- ▶ Commands used with objects in the Java and other environments — for example, `PushButton`, `EditBox`, or `ListBox`.

When either of these kinds of commands refers to a Java object, the command's *recMethod* argument can specify the Java parent object. When specifying a parent object, *recMethod* uses the recognition method `Name=` or either of the following recognition methods:

- ▶ `JavaCaption=$`

The text of the Java window caption. The caption can be used to identify the parent Java object when the object has no programmatic name. The wildcards `?` and `*` are supported. (See *Using Wildcards in Window Captions* on page 4-17.)

This recognition method is used only with window-based parent objects, not with browser-based applets.

- ▶ `JavaClass=$`

The Java class name. The class name can be used to identify the parent Java object when the object has no programmatic name or window caption.

With `JavaObject` and `JavaObjectVP`, `JavaClass=` can also be used to identify the child Java object.

The recognition method qualifier `Index=` can appear after `Name=`, `JavaCaption=`, and `JavaClass=`.

Object Context

For Robot to find the edit boxes, buttons, and other objects that you test, it has to know where to look. For example, if you reference a list view object named `MyList`, Robot needs to know which window the list is in. If you reference a particular item in `MyList`, Robot needs to know both the list that the item is in and the window that the list is in.

Robot locates an object through the object's **context**. Context helps Robot identify an object by providing a point of reference for the object. In other words, the identity of a parent object provides the context for its child objects.

Context for objects is established in either or both of these ways:

- ▶ Through a `Window SetContext`, `Window SetTestContext`, or `Window ResetTestContext` action taken against a particular window.

When context is established in this way, Robot assumes that subsequent actions occur in the specified window until another `Window` command changes the context.

- ▶ Through SQABasic context notation in the *recMethod* argument of a command.

This method establishes context only for the command in which the *recMethod* appears. Subsequent commands are not affected.

The following sections describe these methods of establishing context.

Establishing Context through a Window Command

Robot uses the `Window` command to identify a window as the context for subsequent user actions.

NOTE: In this document, a **window** is a top-level object on the desktop. For example, a dialog box is typically a top-level desktop object.

Suppose you click a push button in the application-under-test during recording. In the script, Robot might describe the action like this:

```
Window SetContext, "Caption=Classics Online", ""
PushButton Click, "Name=cmdOrder"
```

Here's what each line tells Robot:

- ▶ The first line specifies that you took an action in a window. The window is identified by the caption `Classics Online` in the window title bar. The `SetContext` action establishes the specified window as the **current context window** for subsequent user actions.
- ▶ The second line specifies that you clicked a push button with the developer-assigned object name `cmdOrder`. The push button object is assumed to be in the current context window — in this case, the window identified by the caption `Classics Online`.

Robot assumes that the context for subsequent user actions is the current context window. The current context window can (and usually does) change often in a script.

Actions that Set Context

The following *action* argument values for the `Window` command set the context for an object:

`SetContext` – Establishes the current context window for all user action and verification point commands that follow.

`SetTestContext` – Establishes a test context for an object that is outside the scope of the current context window or Object Scripting command. When test context is established for an object, subsequent verification point operations are performed on the specified object until the context changes.

`ResetTestContext` – Restores the context to its state before the last `SetTestContext` action.

See the `Window` user action command in Chapter 6 for more information about `SetContext`, `SetTestContext`, and `ResetTestContext` actions.

Assigning Context to the Currently Active Window

You can assign context to the currently active window without specifically identifying the window. To do so, use the *recMethod* value `CurrentWindow`. For example:

```
Window SetContext, "CurrentWindow", ""
PushButton Click, "Name=cmdOrder"
```

Using Wildcards in Window Captions

If you are using the `Window` command to establish the context window, you can identify the window through its caption. The caption is located in the title bar.

When you specify a window caption, you can type the entire caption, or you can use the following wildcards:

| Wildcard character | Description |
|--------------------|--|
| Question mark (?) | Matches a single character in a caption. |
| Asterisk (*) | Matches any number of caption characters from the asterisk to the next character or, if there are no characters after the asterisk, to the end of the caption. |

When using wildcard characters in a caption, enclose the caption within braces.

Here are some examples of using caption wildcards in the `Window` command to establish context:

```
Window SetContext, "Caption={?otepad}", ""
    ' Matches the window caption "Notepad"

Window SetContext, "Caption={Query*}", ""
    ' Matches any window caption beginning with "Query"

Window SetContext, "Caption={Class*line}", ""
    ' Matches any window caption beginning with "Class" and
    ' ending with "line" (such as "Classics Online")
```

NOTE: Wildcards are not supported in the `Text` recognition method of `DataWindow` and `DataWindowVP` commands.

Using Wildcard Characters as Ordinary Characters

If you want to include a question mark or an asterisk as just another character in a caption rather than as a wildcard, precede the question mark or asterisk with the backslash (\) escape character. Also, to use a backslash as an ordinary character in a caption, precede it with another backslash.

For example, to match the path `c:*.*` in a window caption, use:

```
Caption={c:\\*.*}
```

Alternatively, you could simply omit the braces:

```
Caption=c:\\*.*
```

Establishing Context through Context Notation

Context notation is *recMethod* argument syntax that defines hierarchical relationships between objects. Context notation is used in the *recMethod* argument of user action, verification point, and Object Scripting commands.

In context notation, context for the target object is established by identifying its parent object(s). Note that:

- ▶ Sometimes, the parent object is a window or the desktop. The parent object could also be another object within a window.
- ▶ Sometimes, a child object is actually an item such as a tree view item. These low-level items do not have associated properties, as objects do.

Context notation does not change the current context window. Context notation establishes context only for the command using the context notation in its *recMethod* argument.

With context notation, the *recMethod* argument follows these syntax rules:

- ▶ A backslash (\) between two objects specifies that the first object is the parent of (and the context for) the second object. The backslash is delimited by semicolons (;).

For example, the following code specifies that the tab labeled Album was clicked in a tabbed dialog box:

```
TabControl Click, "Name=tabMain;\;ItemText=Album", ""
```

In this example, the item Data.mdb was clicked in a list view object:

```
ListView Click, "ObjectIndex=1;\;ItemText=Data.mdb", "Coords=10,8"
```

This type of context notation is used with hierarchical objects (such as list view and tree view) and with Object Scripting commands.

- ▶ A backslash at the beginning of a recognition method specifies that the next object in the path is a child of the desktop. The backslash is followed by a semicolon (;).

This example shows a *recMethod* argument that specifies a path from the desktop to the target object:

```
"\;Type=Window;Caption=Notepad;\;Type=EditBox;ObjectIndex=1"
```

This type of context notation is used only with Object Scripting commands.

- ▶ A dot-backslash (.\) represents the current context window. If the path includes an object after the dot-backslash, the dot-backslash is followed by a semicolon (;\).

In this example, Robot retrieves the recognition string for the current context window:

```
Result = SQAGetProperty (".\", "Recognition", value)
```

In this example, Robot retrieves the number of rows in the grid `myGrid`, which is in the current context window:

```
Result = SQAGetProperty (".\;Name=myGrid", "Rows", value)
```

This type of context notation is used only with Object Scripting commands.

- ▶ Backslash and dot-backslash characters are delimited by semicolons (;).
- ▶ In addition, with user action and verification point commands, use the *recMethod* value `ChildWindow` when specifying an MDI window. In this example, `Book2` is shown to be a child window of Microsoft Excel:

```
Window SetContext, "Caption=Microsoft Excel", ""
Window WMinimize, "Caption=Book2;ChildWindow", ""
```

NOTE: Multi-object recognition method paths can be difficult to construct. To be sure you define the correct recognition method for an object, record a temporary script and click on the object. Robot will find the correct recognition method for you. You can then copy the recognition method into your own script. For more information, including information about finding recognition method information programmatically, see the section *Getting Help Defining Recognition Methods* in Chapter 5, *Enhancements to Recorded Scripts*.

Using Wildcards in Window Captions

If you are establishing a window as the context for a child object, you can identify the window through its caption. The caption is located in the title bar.

When you specify a window caption, you can type the entire caption, or you can use the following wildcards:

| Wildcard character | Description |
|--------------------|--|
| Question mark (?) | Matches a single character in a caption. |
| Asterisk (*) | Matches any number of caption characters from the asterisk to the next character or, if there are no characters after the asterisk, to the end of the caption. |

When using wildcard characters in a caption, enclose the caption within braces.

Customizing Scripts

Here are some examples of using caption wildcards in the *recMethod* argument of an Object Scripting command:

```
"\;Type=Window;Caption={?otepad};\;Type=EditBox;ObjectIndex=1"  
  ' Matches the window caption "Notepad"  
  
"\;Type=Window;Caption={Query*};\;Type=EditBox;ObjectIndex=1"  
  ' Matches any window caption beginning with "Query"  
  
"\;Type=Window;Caption={Class*line};\;Type=PushButton;ObjectIndex=1"  
  ' Matches any window caption beginning with "Class" and  
  ' ending with "line" (such as "Classics Online")
```

Using Wildcard Characters as Ordinary Characters

If you want to include a question mark or an asterisk as just another character in a caption rather than as a wildcard, precede the question mark or asterisk with the backslash (\) escape character. Also, to use a backslash as an ordinary character in a caption, precede it with another backslash.

For example, to match the path `c:*.*` in a window caption, use:

```
Caption={c:\\*.*}
```

Alternatively, you could simply omit the braces:

```
Caption=c:\\*.*
```

Default Context

Object context has different defaults in different situations:

- ▶ The default context for a window is the desktop.
- ▶ The default context for other objects is the context set through the most recent `SetContext`, `SetTestContext`, or `ResetTestContext` action.

Customizing Scripts

The SQABasic scripting language gives you much of the programming flexibility of Microsoft Basic and other programming languages. For example, you can:

- ▶ Edit the scripts that Robot automatically generates.
- ▶ Add new commands, variables, and constants to scripts.
- ▶ Create custom sub procedures and functions for a script.
- ▶ Create library files for sub procedures and functions called from multiple scripts.
- ▶ Declare variables, constants, functions, and sub procedures in header files.
- ▶ Create a script template.

Script Editing Basics

To edit a script in Robot:

1. Click File → Open → Script.
2. Select the script to edit.
3. Click OK.

You can edit the SQABasic commands that Robot generates during recording, and you can add new commands. Add and edit commands according to the syntax descriptions in Chapter 6, *Command Reference*.

Declaring Variables and Constants

Declaring variables and constants is a fundamental script editing task you perform when editing a script. The following sections describe local, module-level, and global declarations of variables and constants.

Declaring Local Variables and Constants

You can declare local variables in a script or library source file.

The scope of local variables and constants is confined to the procedure in which the declarations appear.

You can insert a local declaration of a variable or constant anywhere within a procedure, as long as the declaration appears before its first use. Typically, however, variable and constant declarations appear at the beginning of the procedure.

Use `Dim` to declare a variable and `Const` to declare a constant.

In the following example, the variables `Result` and `value`, and the constant `TESTID`, are local to the `Main` sub procedure. Other procedures that may exist in this script file cannot access `Result`, `value`, or `TESTID`.

```
Sub Main
  Dim Result As Integer
  Dim value As String
  Const TESTID As String = "Test Plan Alpha: "
  . . . ' Continue processing Main sub procedure
End Sub
```

Declaring Module-Level Variables and Constants

A module is an SQABasic script or library source file.

If you declare module-level variables and constants inside a script or library file, their scope spans all the sub procedures and functions in that file.

Module-level variable and constant declarations appear at the beginning of the file, above the `Main` sub procedure (for scripts) and any other procedures in the file. Use `Dim` to declare a module-level variable and `Const` to declare a module-level constant.

In this example, the variable `value` and the constant `TESTID` can be accessed by all the procedures in the script file. The variable `Result`, however, is local to the `Main` sub procedure.

```
Dim value As String
Const TESTID As String = "Test Plan Alpha: "

Sub Main
    Dim Result As Integer
    . . .
    ' Continue processing Main sub procedure
End Sub
```

NOTE: For information about declaring variables and constants that are available to any module, see *Using SQABasic Header Files* on page 4-29.

Declaring Global Variables and Constants

If you declare a global variable or constant in a module, the variable or constant is validated at module load time:

- ▶ **Variables.** If you attempt to load a module that has a global variable declared, and the variable has a different data type than an existing global variable of the same name, the module load fails.
- ▶ **Constants.** If a declared constant has already been added to the runtime global area, the constant's type and value are compared to the previous definition, and the load fails if a mismatch is found. This is useful as a mechanism for detecting version mismatches between modules.

A definition for each global constant is stored in every compiled module. Other constants are only stored in a module if they are referenced by the module

Because global variables and constants have the potential to make modules large and slow, you should declare global variables and constants only when necessary.

The following table shows the difference between local or module-level declarations and global declarations:

| Local or module-level declaration | Global declaration |
|--|---|
| <code>Dim myVariable as Integer</code> | <code>Global myVariable as Integer</code> |
| <code>Const MYCONSTANT as String = "aa"</code> | <code>Global Const MYCONSTANT as String = "aa"</code> |

You can also declare global variables and constants in a header file.

Adding Custom Procedures to a Script

You can write custom sub procedures and functions and add them to the `Main` sub procedure that Robot generates in a script. If you add a custom sub procedure or function to a script, you can call it from `Main` or other procedures in the script.

For information about defining procedures in a script, see the following sections of Chapter 6, *Command Reference*:

- ▶ `Sub...End Sub` to define a sub procedure
- ▶ `Function...End Function` to define a function

Declaring a Procedure Residing in a Script

Procedure declarations typically appear at the beginning of a file, before the first `Sub...` or `Function...` statement in the file. Procedure declarations cannot appear within a procedure's `Sub...End Sub` or `Function...End Function` statements.

However, you can insert a procedure declaration anywhere within a file, as long as the declaration appears before its first use and does not appear within a procedure.

Use the `Declare` statement to declare procedures.

NOTE: For information about declaring custom procedures that are available to any module, see *Using SQABasic Header Files* on page 4-29.

Declaring a Sub Procedure

Here is an example of declaring a sub procedure named MySub. MySub has a string argument and an integer argument:

```
Declare Sub MySub(arg1 As String, arg2 As Integer)

Sub Main
  Dim s As String
  Dim i As Integer
  . . .
  Call MySub(s,i)
  . . .
End Sub

Sub MySub(arg1 As String, arg2 As Integer)
  . . . ' Process the passed values
End Sub
```

Declaring a Function

Here is an example of declaring a function named MyFunc. MyFunc has a string argument and an integer argument. It also returns a status code as a string:

```
Declare Function MyFunc(arg1 As String, arg2 As Integer) As String

Sub Main
  Dim s As String
  Dim i As Integer
  Dim status As String
  . . .
  status=MyFunc(s,i)
  If status = "Success" Then
    . . .
  End If
  . . .
End Sub

Function MyFunc(arg1 As String, arg2 As Integer) As String
  . . . ' Process the passed values
  MyFunc="Success"
End Function
```

Using a Procedure Definition as a Declaration

A procedure definition also serves as a declaration. As a result, procedure declarations are not always required. For instance, in the previous example, if the order of the procedures is reversed, no declaration is needed for MyFunc:

```
Function MyFunc(arg1 As String, arg2 As Integer) As String
    . . . ' Process the passed values
    MyFunc="Success"
End Function

Sub Main
    Dim s As String
    Dim i As Integer
    dim status As String
    . . .
    status=MyFunc(s,i)
    If status = "Success" Then
        . . .
    End If
End Sub
```

Example of a Custom Procedure

In the following example, the custom function MyProp is added to the script file DB5. MyProp gets information about a property by calling SQAGetProperty, and reports any SQAGetProperty errors to the log as warnings.

The calling procedure, Main, expects a window entitled Make An Order to be the currently active window. If it isn't the active window, Main makes it the active window and reports an error to the log. Main then performs an object property verification point on the window's Order button.

```
Dim value as Variant ' Retrieved in SQAGetProperty
Const TESTID As String = "Test Plan Alpha: "
Declare Function GetProp(recMethod As String, property As String) As Integer

Sub Main
    Dim Result As Integer

    'Initially Recorded: 06/22/98 14:00:24
    'Script Name: DB5
    Result= GetProp("CurrentWindow","Caption")
    If Result = sqaSuccess Then
        If value <> "Make An Order" Then
            AppActivate "Make An Order"
            SQAErrorMessage SQAFail, TESTID + value + " is the active window", ""
        End If
        Window SetTestContext, "Name=frmOrder", ""
        Result = PushButtonVP (CompareProperties, "Name=cmdOrder", "VP=OrdBtn")
        Window ResetTestContext, "", ""
    End If
End Sub

Function GetProp(recMethod As String, property As String) As Integer
    Dim retVal as Integer

    retVal=SQAGetProperty(recMethod,property,value)
    GetProp=retVal
    If retVal <> sqaSuccess Then
        SQAErrorMessage SQAWarning, TESTID + Error$(retVal), ""
    End If
End Function
```

Adding Custom Procedures to a Library File

A library file contains one or more sub procedures and functions that are called from procedures in other files.

SQABasic supports these kinds of library files:

- ▶ SQABasic library files. SQABasic library source files can have either a .sbl or .rec extension. Compiled SQABasic library files have the extension .sbx.

Note that .rec files can be used as script files or as library files, but .sbl files can only be used as library files.

- ▶ Dynamic-link library files (extension .dll).

The following table summarizes the differences between library files:

| | .sbl | .rec | .dll |
|------------------------|---|---|--|
| Location | Repository (folder SQABas32) | Project (folder Script) | User assigned |
| Scope | Available to all files in the repository | Available to all files in the project | Depends on location |
| Verification points | No support | Supports all standard Robot verification points | Supports custom verification points |

Any .rec file can be used as a library file. However, if a .rec file is also to be used as a script (that is, if it is to be executable directly from Robot or from the `CallScript` command), it must have a `Main` sub procedure.

To see a working example of a library file, open the Rational Robot Help and search the index for *library source files*.

The following sections describe how to work with library files.

Working With SQABasic Library Files

Adding custom procedures to an SQABasic library file is the same as adding custom procedures to a script. For information, see the following sections of Chapter 6, *Command Reference*.

- ▶ `Sub . . . End Sub` to define a sub procedure
- ▶ `Function . . . End Function` to define a function

Creating SQABasic Library Files

To create a new .sbl library file:

1. In Robot, click File → New → SQABasic File.
2. Click Library Source File, and then click OK.

You name the file (or accept the default name) the first time you save it.

A library file cannot have the same name as the script file that calls it. For instance, myscript.rec cannot call a function in myscript.sbl.

NOTE: For your convenience, Robot provides a blank library source file called global.sbl. You can add your custom procedures to this file and/or create new library source files. To open this file in Robot, click File → Open → SQABasic File, select global.sbl, and then click Open.

To create a new .rec library file:

1. In Robot, click File → New → Script.
2. Type the name of the file to create and optionally, a description.
3. Click the file type GUI if it is not already selected.
4. Click OK.

Editing SQABasic Library Files

To open an existing .sbl library file:

1. In Robot, click File → Open → SQABasic File.
2. In Files of type, select Library Source Files (*.sbl).
3. Click the file to edit, and then click Open.

To open an existing .rec library file:

1. In Robot, click File → Open → Script.
2. Click the name of the file to edit, and then click OK.

Compiling SQABasic Library Files

Compile the SQABasic library file before you attempt to access it at test runtime.



Compiling SQABasic library files is the same for both .sbl files and .rec files. The fastest way to compile is to click the Compile button on the Robot toolbar. Compiling the file also saves it.

Compiled .sbl and .rec library files have the extension .sbx.

Declaring a Procedure Residing in an SQABasic Library File

If a custom procedure is in an SQABasic library file, you declare the library file in the same `Declare` statement you use to declare the procedure.

Here is an example of a declaration of a custom procedure (MySub) and an SQABasic library file (MyLib):

```
Declare Sub MySub BasicLib "MyLib" (arg1 As String, arg2 As Integer)
```

Note the differences (shown in bold type) between this procedure declaration and the module-level procedure declaration example on page 4-24:

- ▶ The word `BasicLib` is added to the declaration, indicating that the declared procedure MySub is in an SQABasic library file.
- ▶ The name of the library file (MyLib), in quote marks, follows the `BasicLib` designation.

Because the `BasicLib` keyword indicates that a `.sbx` library file (as opposed to a `.dll` library file) is being declared, the `.sbx` extension in the declaration is not required or recommended.

Where to Declare an SQABasic Library File

You can declare an SQABasic library file in any of these locations:

- ▶ In a script or other library file, for use by the procedures in that module only
- ▶ In a header file, for use by any module that references the header file

Working With DLL Files

SQABasic procedures can call procedures stored in DLL files. For example, they can call the procedures stored in Microsoft Windows DLLs such as `Kernel32.dll`.

Robot does not provide a tool for creating DLLs. To add procedures to a DLL file, you need a tool such as Microsoft Visual C++ or Visual Basic.

Declaring a Procedure Residing in a DLL File

If a procedure is in a DLL file, you declare the DLL file in the same `Declare` statement you use to declare the procedure.

Here is an example of a declaration of a custom procedure (MySub) and a DLL file (MyDLL):

```
Declare Sub MySub Lib "MyDLL" (ByVal arg1 As String, ByVal arg2 As Integer)
```

Note the differences (shown in bold type) between this procedure declaration and the module-level procedure declaration example on page 4-24:

- ▶ The word **Lib** is added to the declaration, indicating that the declared procedure **MySub** is in a **.dll** library file (as opposed to a **.sbl** or **.rec** SQABasic library file).
- ▶ The name of the library file (**MyDLL**), in quote marks, follows the **Lib** designation.
- ▶ Argument declarations include the keyword **ByVal**. For information about using the keyword **ByVal** (or **Any**) with argument declarations for DLL procedures, see the **Declare** statement in Chapter 6, *Command Reference*.

If the compiled library file (**.dll**) is located in the repository's **SQABas32** directory or in the system path, you don't need to specify a path in the declaration. If the library file is not in **SQABas32** or in the system path, you do need to specify the path — for example:

```
Declare Sub MySub Lib "E:\MyDLL" (ByVal arg1 As String, ByVal arg2 As Integer)
```

Where to Declare a DLL File

You can declare a DLL file in any of these locations:

- ▶ In a script or SQABasic library file, for use by the procedures in that module only
- ▶ In a header file, for use by any module that references the header file

Using SQABasic Header Files

An SQABasic header file contains a list of declarations. You can use header files to declare constants, variables, custom sub procedures, and custom functions.

The declarations in a header file apply to any module (script or library file) that references the header file. Use `' $Include` to reference a header file.

SQABasic supports two types of header files:

- ▶ Header files are stored in the **SQABas32** folder of the repository. They can be accessed by all modules within the repository.
- ▶ Project header files are stored in the **Script** folder of the project. They can be accessed by all modules within the project.

Both types of SQABasic header files have the extension **.sbh**.

To see a working example of a header file, open the Rational Robot Help and search the index for *header files*.

Creating and Editing a Repository-Wide Header File

To create a new header file that can be accessed by any module in the repository:

1. In Robot, click File → New → SQABasic File.
2. Click Header File, and then click OK.

You name the file (or accept the default name) the first time you save it.

NOTE: For your convenience, Robot provides a blank repository-wide header file called `global.sbh`. You can add your global declarations to this file and/or create new header files. To open this file in Robot, click File → Open → SQABasic File, select `global.sbh`, and then click Open.

To edit an existing repository-wide header file:

1. In Robot, click File → Open → SQABasic File.
2. In Files of type, select Header Files (*.sbh).
3. Click the file to edit, and then click Open.

Creating and Editing a Project Header File

To create a new project header file:

- ▶ In Robot, click File → New → Project Header File.

You name the file (or accept the default name) the first time you save it.

To edit an existing project header file:

1. In Robot, click File → Open → Project Header File.
2. Click the file to edit, and then click Open.

Saving SQABasic Header Files

After you add declarations to an SQABasic header file, save the file. When you create or edit an SQABasic header file, save it before you compile a script or library file that references the SQABasic header file. You don't compile SQABasic header files.

Scope of Declarations in SQABasic Header Files

At compile time, the `'$Include` command logically inserts the SQABasic header file declarations into the script at the line where the `'$Include` command is located (*logically*, because the script is not physically changed).

As a result, the scope of the declarations in the SQABasic header file is determined, in part, by the location of the `'$Include` command. When the `'$Include` command is located before the first procedure in the module, the SQABasic header file declarations apply to all the procedures in the module.

Scope is also determined by whether you declare a variable or constant as global. For more information, see *Declaring Global Variables and Constants* on page 4-22.

Declaring Global Variables and Constants Inside Header Files

You can declare global variables and constants inside an SQABasic header file, just as you can declare them inside a module. For information, see *Declaring Global Variables and Constants* on page 4-22.

Declaring Global Procedures inside Header Files

You declare sub procedures and functions in an SQABasic header file exactly as you declare them in a script:

- ▶ For information about declaring procedures that reside in a script file, see *Declaring a Procedure Residing in a Script* on page 4-23.
- ▶ For information about declaring procedures that reside in an SQABasic library file, see *Declaring a Procedure Residing in an SQABasic Library File* on page 4-28.
- ▶ For information about declaring procedures that reside in a DLL file, see *Declaring a Procedure Residing in a DLL File* on page 4-28.

Referencing an SQABasic Header File

For the procedures in a script or SQABasic library file to be able to use the variables, constants, and procedures declared in an SQABasic header file, the script or library file needs to reference the header file. You reference a header file through the `'$Include` command.

To have header file declarations apply to all the procedures in a module, place `'$Include` at the beginning of the module — for example:

```
'$Include "global.sbh"

Sub Main
    .
    .
End Sub
```

Customizing Scripts

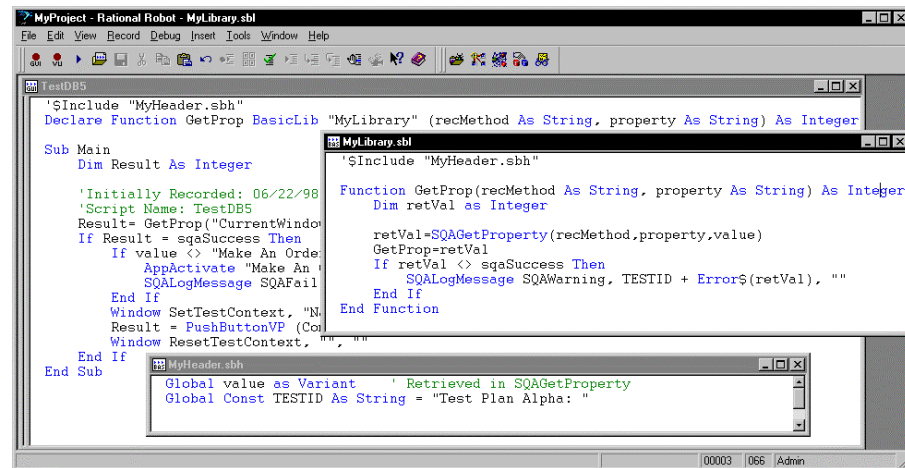
Note that:

- ▶ The SQABasic header file name is enclosed in double quotation marks (").
- ▶ SQABasic header file names are not case sensitive.
- ▶ If a repository-wide header file resides in the default location SQABas32, or a project header file resides in the default Script folder for the Project, no path is necessary in the '\$Include command. If an SQABasic header file is located in any other folder, specify the full path.

The '\$Include command begins with a single quotation mark ('), which normally indicates a comment. But when a single quotation mark is followed by a dollar sign (\$), a special SQABasic command is indicated.

Sample Library and Header Files

The following figure contains the same code as the script DB5 on page 4-25. But now, the variable and constant declarations have been moved to the global header file MyHeader.sbh, and the custom procedure has been moved to the library file MyLibrary.sbl.



```
MyProject - Rational Robot - MyLibrary.sbl
File Edit View Record Debug Insert Tools Window Help

TestDB5
'$Include "MyHeader.sbh"
Declare Function GetProp BasicLib "MyLibrary" (recMethod As String, property As String) As Integer

Sub Main
  Dim Result As Integer
  'Initially Recorded: 06/22/98
  'Script Name: TestDB5
  Result = GetProp("CurrentWindow")
  If Result = sqaSuccess Then
    If value <> "Make An Order" Then
      AppActivate "Make An Order"
      SQAErrorMessage SQAFail
    End If
    Window SetTestContext, "N..."
    Result = PushButtonVP (ControlID)
    Window ResetTestContext, "N..."
  End If
End Sub

MyLibrary.sbl
'$Include "MyHeader.sbh"
Function GetProp(recMethod As String, property As String) As Integer
  Dim retVal As Integer
  retVal = SQAGetProperty(recMethod, property, value)
  GetProp = retVal
  If retVal <> sqaSuccess Then
    SQAErrorMessage SQAWarning, TESTID + Error$(retVal), ""
  End If
End Function

MyHeader.sbh
Global value as Variant ' Retrieved in SQAGetProperty
Global Const TESTID As String = "Test Plan Alpha: "
```

Note that:

- ▶ The variable and constant declarations in the header file have a different syntax than they did when declared inside the script.
- ▶ The declaration of the function `GetProp` now includes the fact that it resides within an SQABasic library (through the keyword `BasicLib`). The declaration also specifies the name of the compiled library (`MyLibrary`).
- ▶ For the script `TestDB5` and the library file `MyLibrary` to access the same variables and constants, both files `'$Include` the header file `MyHeader.sbh`, where the variable and constant declarations reside.
- ▶ Because the custom procedure `GetProp` is declared inside the script `DB5`, it can be called by all procedures (such as `Main`) in that script.

`GetProp` can also be declared in a header file, so that procedures in any script can call it. However, `GetProp` cannot be declared in the header file `MyHeader.sbh`, because the library where `GetProp` resides (`MyLibrary`) references that header file. A library file cannot `'$Include` a header file that contains a declaration of a procedure residing within that library file.

If the declaration of `GetProp` resided in a header file named `MyProcs.sbh`, this is how the script `TestDB5` would begin:

```
'$Include "MyHeader.sbh"
'$Include "MyProcs.sbh"

Sub Main
    . . .
End Sub
```

Using the Template File

The template file `testproc.tpl` lets you automatically add entries such as comments and include statements to new scripts. Any text you add to the `testproc.tpl` file is automatically added to each new script you create.

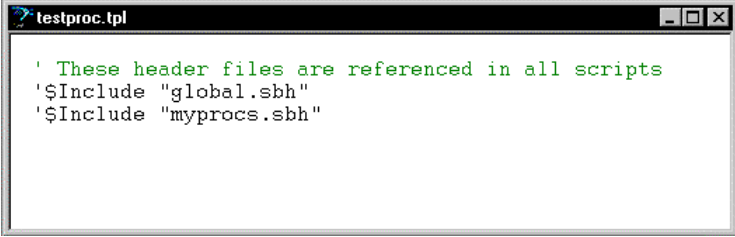
Template entries are only added to new scripts. They are not added to new library files or header files.

To edit the `testproc.tpl` template file:

1. In Robot, click `File` → `Open` → `SQABasic File`.
2. In `Files of type`, select `Template Files (*.tpl)`.
3. Select `testproc.tpl`, and then click `Open`.

Customizing Scripts

4. Define the template entries you want — for example:



```
' These header files are referenced in all scripts
'$Include "global.sbh"
'$Include "myprocs.sbh"
```

5. Click File → Save.
6. Click File → Close.

▶ ▶ ▶ C H A P T E R 5

Enhancements to Recorded Scripts

During recording, Rational Robot automatically generates most of the activities that you will need a script to perform. However, there are some activities that Robot does not generate during recording. These activities include:

- ▶ Object scripting
- ▶ Managing custom verification points
- ▶ Comparing environment states
- ▶ Displaying messages in Robot
- ▶ Using datapools
- ▶ Accessing external applications

Object Scripting

SQABasic's powerful **Object Scripting** commands let you access an application's objects and object properties from within a script. The tasks you can perform with Object Scripting commands include retrieving and setting an object's properties. For example, you could use the `SQAGetProperty` command to retrieve properties such as the height, location, or value of an edit box.

You can also perform other kinds of tasks with Object Scripting commands, such as executing a method associated with an object, and checking to see if an object exists before performing actions against the object.

Object Scripting commands can only be inserted by manually editing the script. Robot does not generate these commands during recording.

See *Object Scripting Commands* in Chapter 2, *Functional List*, for a summary of each Object Scripting command.

Specifying an Object

You specify the object you want to access through a recognition string in the recognition method (*recMethod*) argument of an Object Scripting command.

The recognition method values you use to identify an object depend on the object you're accessing. For example, if you're accessing a push button object, use the recognition method values listed for the `PushButton` user action command. (See the description of the `PushButton` command in Chapter 6, *Command Reference*.)

In addition, you might need to specify one or both of the following kinds of information to uniquely identify an object for an Object Scripting command:

- ▶ Object type
- ▶ Object context

Object Type

With Object Scripting commands, just as with user action and verification point commands, the recognition method argument uniquely identifies the object to be accessed. However, where the object *type* is implicit in the specific user action or verification point command name itself, you sometimes have to explicitly define the object type in Object Scripting commands.

For example, suppose you record a mouse click on an OK push button. Robot records the user action with this command:

```
PushButton Click, "Text=OK"
```

The object type, a push button, is made clear from the command name itself.

But suppose you want to determine whether the OK button's `Enabled` property is set to `True` or `False`. If you call the Object Scripting command `SQAGetProperty` to retrieve this information, and the command uses the same *recMethod* value that the above `PushButton` command used, this is the way the new command looks:

```
Result=SQAGetProperty("Text=OK", "Enabled", value)
```

Nowhere in this command is the object type — a push button — specified. If no other object on the current context window contains the text `OK`, there is no confusion about the object you're accessing. But if another object uses the same label as the push button (for example, a check box with the caption `OK`), the command can't be sure which object you want and may retrieve the wrong value.

To be sure that you uniquely identify the object you want to access, include the object type in the *recMethod* argument, as follows:

```
Result=SQAGetProperty("Type=PushButton;Text=OK", "Enabled", value)
```

SQABasic Object Type Names

The table below lists the valid object types you can specify in the *recMethod* argument of an Object Scripting command.

The names may not be exactly the same as the names used in the development environment. (For example, the SQABasic object type ComboBox may be called DropDownList in the development environment.)

Object type names are not case sensitive.

Note that some development environments offer special object types beyond those available to all development environments.

Here is the table of valid *recMethod* object types:

| Development environment | Valid values for Type= in <i>recMethod</i> | |
|---|--|--|
| Microsoft Windows objects available to all development environments | AnimateControl Calendar CheckBox ComboBox ComboEditBox ComboListBox DateTime Desktop EditBox Generic GroupBox HDItem Header HotKeyControl Image IPAddress Label ListBox ListView | LVItem Pager ProgressBar PushButton RadioButton Rebar RichEdit ScrollBar SpinControl StatusBar TabControl TBItem TCItem Toolbar Trackbar TreeView TVItem Window |
| HTML | HTML HTMLActiveX HTMLDocument HTMLHidden | HTMLImage HTMLLink HTMLTable |

▶ ▶ ▶

Object Scripting

▶ ▶ ▶

| Development environment | Valid values for Type= in <i>recMethod</i> | |
|-------------------------|--|---|
| Java | JavaCanvas JavaListView JavaMenu JavaObject JavaPanel JavaPopupMenu | JavaSplitPane JavaSplitter JavaTable JavaTableHeader JavaTree JavaWindow |
| Oracle | Block Canvas ChartItem DisplayItem Form Image | LOV* OLEContainer RadioGroup RecordGroup* UserArea |
| PeopleSoft | Border Calendar Field Frame Image LongEdit PSCalendar PSColumn PSGrid PSGridHeader PSMapItem PSMenu | PSNavigator PSPanel PSSpin PSTimeSpin PSTree PSTreeItem PSTreeHeader SecondaryPanel SubPanel StaticImage Text |
| PowerBuilder | DataWindow DropDownDataWindow DropDownListBox DWBitmap DWColumn DWComputedField DWEllipse DWGraph | DWLine DWOLE DWRectangle DWReport DWRoundRectangle DWTableBlob DWText |
| Visual Basic | Image Line | OLE Shape |

* This object can only be accessed through Object Scripting commands. It can't be accessed when you record user actions or verification points with Robot.

NOTE: Developers assign a name to an object to uniquely identify the object in the development environment. Because object names are usually unique, you typically can use `Name=` to identify an object without using `Type=`.

Object Context

By default, the context for an object you specify in the recognition method argument is the current context window. For example, the following push button object is assumed to be in the current context window (*recMethod* is the first argument in the `SQAGetProperty` command):

```
Result=SQAGetProperty("Type=PushButton;Text=OK", "Enabled", value)
```

If the object you want to access isn't a direct child of the current context window, or if you want to define a full object path for the Object Scripting command, you define the context through context notation, as described in the section *Establishing Context through Context Notation* in Chapter 4, *SQABasic Scripts*.

For example, the following two code fragments each access a combo box object in a window whose caption is Make An Order. The first `SQAGetProperty` example uses the current context window:

```
Window SetContext, "Caption=Make An Order", ""
Result=SQAGetProperty("Type=ComboBox;Name=cmbCardType", "Text", value)
```

The second `SQAGetProperty` example uses context notation to establish context:

```
Result=SQAGetProperty("\;Caption=Make An Order;\;Type=ComboBox;
Name=cmbCardType", "Text", value)
```

Remember, context notation assigns context *locally* — it only affects the command in which the context notation appears. Context notation does not change the current context window.

Other Ways to Specify an Object

The following *recMethod* values are useful when you don't know the name of the object you want to access:

CurrentWindow – Specifies the currently active window as the window object to access. For example, the following command retrieves the text displayed in the title bar of the currently active window:

```
Result=SQAGetProperty("CurrentWindow", "Caption", value)
```

CurrentFocus – Specifies the object that currently has the Windows focus as the object to access. For example, the following command retrieves the height of the object with the Windows focus:

```
Result=SQAGetProperty("CurrentFocus", "Height", value)
```

Specifying the Object Property

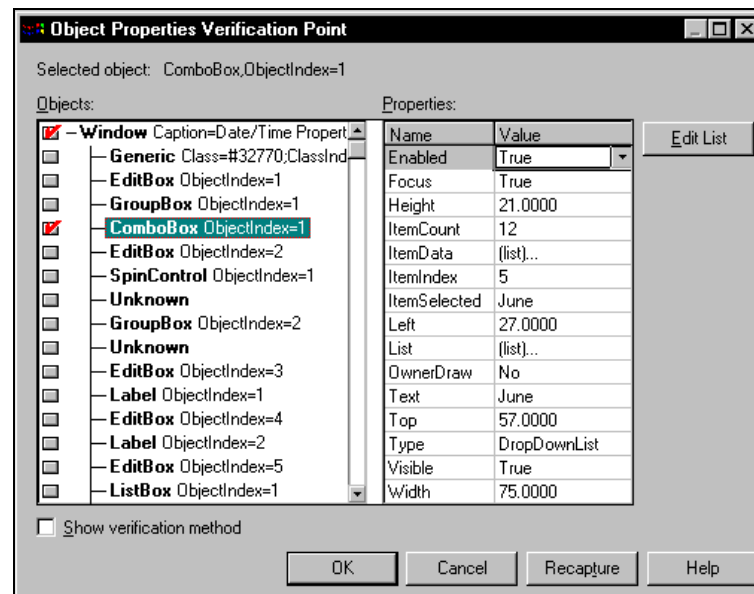
To specify a property to access with an Object Scripting command, assign the property name to the command's *property* argument.

The following sections describe how you can find out which properties you can access through an Object Scripting command.

Properties Assigned in the Development Environment

The properties you can access for a given object include the properties you can define for the object in the development environment.

These are the same properties you see when you perform an Object Properties verification point for the object. For example, suppose you capture verification point information for the Months field object of the Windows NT Date/Time Properties dialog box. These are the object properties you see listed on the Object Properties Verification Point dialog box:



The properties you can access for the Month field (a combo box) are listed in the Name column in the preceding figure. To specify a property to access in an Object Scripting command, insert the property name in the *property* argument of the Object Scripting command you're using.

NOTE: Property names are case sensitive. Names must be typed exactly as listed in the Name column of the Object Properties Verification Point dialog box.

Here's an example of how to use the `SQAGetProperty` command to retrieve the current value of the Month field on the Windows NT Date/Time Properties dialog box. The *property* argument is in bold type:

```
Sub Main
  Dim Result As Integer
  Dim value as Variant
  Window SetContext, "Caption=Date/Time Properties", ""
  Result=SQAGetProperty("Type=ComboBox;ObjectIndex=1", "Text", value)
  MsgBox "Current month is " + value
End Sub
```

To display this dialog box before running the script, double-click the date in the Windows NT taskbar.

Additional Properties

In addition to the properties that are captured when you record an Object Properties verification point for a given object, you can access the following properties for any object:

| Property | Description |
|-----------------|---|
| Class | The object's class name. |
| ClientRect | The coordinates of the object, in pixels, relative to the client area of the window (in the format "x1,y1 to x2,y2"). |
| Environment | The name of the development environment (such as Visual Basic or PowerBuilder) in which the object was created. |
| FullRecognition | A full-path recognition string that identifies the object and all its parent objects up to the desktop. |
| hWnd | The window handle, if any, associated with an object. |
| ModuleFileName | The full path and file name of the library file or executable file that controls the specified object. For example, <code>ModuleFileName</code> could be: The application's executable file name (as is often the case for top-level windows). A .DLL (for example, objects within a standard File Open dialog box may have a <code>ModuleFileName</code> of <code>C:\WIN95\SYSTEM\COMDLG32.DLL</code> , which is the common dialog box library). |

► ► ►

▶ ▶ ▶

| Property | Description |
|-------------------|--|
| Name | The object name that is assigned in the development environment. |
| ObjectType | The SQABasic name for the object's type. For a list of the object names that SQABasic supports, see the table beginning on page 5-3. |
| ParentRecognition | A full-path recognition string that uniquely identifies the object's immediate parent. |
| Recognition | A recognition string that uniquely identifies the object within its parent. |
| ScreenRect | The coordinates of the object, in pixels, relative to the screen (in the format "x1,y1 to x2,y2"). |

NOTE: Because a `Rect` can't be stored as a `Variant`, you can't use `SQAGetProperty` to retrieve a value for the `ClientRect` or `ScreenRect` property. Instead, use `SQAGetPropertyAsString` to retrieve the value in `String` form ("x1,y1 to x2,y2").

Array of Property Values

Some property values are stored as arrays — for example, the list of items stored in a combo box control.

Specifying Individual Elements in an Array

You use standard SQABasic array notation to access the elements in an array of property values. For example, in the following code, the `property` argument (argument 2) shows how to specify the third item in a combo box:

```
SQAGetPropertyAsString "Type=ComboBox;ObjectIndex=1", "List(2)", item
```

Note that the array is 0-based. Indices to arrays of property values are almost always 0-based. The only exceptions are some Visual Basic or OCX/ActiveX controls where the array has been specifically declared as 1-based.

Because 1-based arrays of property values are rare, assume that the array you're accessing is 0-based. If you have a problem accessing an OCX/ActiveX array, consult the documentation for the OCX/ActiveX control to find out how the array is indexed.

Retrieving an Entire Array

You can retrieve the entire array of values for a property by calling either of these commands:

- ▶ `SQAGetPropertyArray`
- ▶ `SQAGetPropertyArrayAsString`

These commands return values as a Basic array which is always 0-based.

NOTE: `SQAGetProperty` and `SQAGetPropertyAsString` retrieve just a single element in an array. If you use these commands to try to retrieve an entire array (by not specifying an index value in the *property* argument), the error `sqaArraysNotSupported` is returned.

Retrieving the Number of Elements in an Array

If you want to retrieve the number of elements in an array, use the command `SQAGetPropertyArraySize`.

Getting Help Defining Recognition Methods

When specifying the object to access, you have to uniquely identify the object in the *recMethod* argument of the Object Scripting command.

Multi-object recognition method paths can be difficult to construct. The following sections describe two ways you can get help in defining recognition method values:

- ▶ Letting Robot define recognition method values for you
- ▶ Finding recognition method values programmatically

Letting Robot Define Recognition Method Values

In many cases, Robot can define recognition method values for you. To have Robot do so, perform these steps:

1. Record a temporary script, click on the object you want to define a recognition method for, and then stop recording.
2. Copy the recorded recognition method.
3. Open your own script and paste the recognition method into the *recMethod* argument of the appropriate command.

When using this method, keep the following points in mind:

- ▶ Make sure that the context window is the same for the command in your script and the `Click` action you recorded in the temporary script.

For information about the context window, see the section *Establishing Context through a Window Command* in Chapter 4, *SQABasic Scripts*.

- ▶ If you are defining a recognition method for an Object Scripting command, and the above method doesn't work, you might need to add a `Type=` value to the recognition method.

For information about `Type=` values, see the section *Object Type* on page 5-2.

Finding Recognition Method Values Programmatically

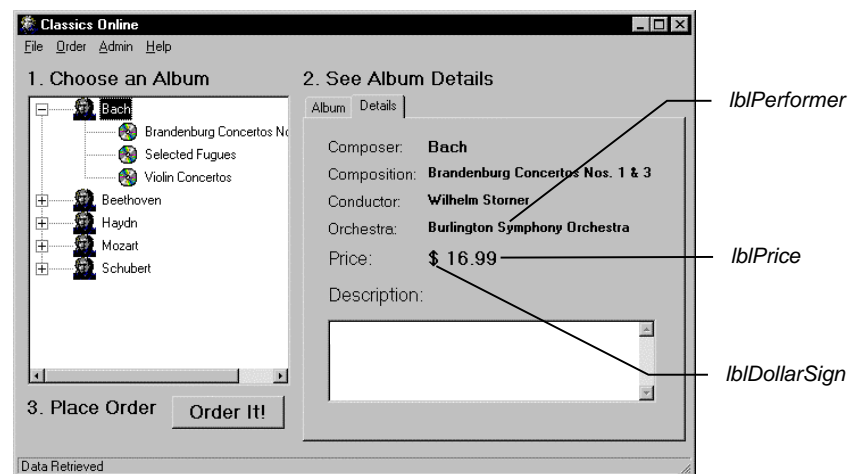
The following Object Scripting commands may be useful if you need to construct a recognition method path programmatically within your script:

- ▶ `SQAGetProperty` or `SQAGetPropertyAsString`, when used to retrieve a `Recognition`, `ParentRecognition`, or `FullRecognition` property. These properties are listed in the table on page 5-7.
- ▶ `SQAGetChildren`.

When you retrieve recognition methods through these commands, a `Type=` object definition is included in all returned values.

Examples

All of the following examples are in the context of the Classics Online window (Name=`frmMain`) shown below. The clicked item is Bach (`ItemText=Bach`), an item in the tree view object (Name=`treMain`):



- ▶ To find the recognition method of the currently active window:

```
Result=SQAGetProperty( ".\" , "Recognition" , value)
```

Returned value:

```
Type=Window;Name=frmMain
```

- ▶ To find the immediate parent of the tree view item Bach:

```
Result=SQAGetProperty( "Name=treMain;\ ;ItemText=Bach" ,  
"ParentRecognition" , value)
```

Returned value:

```
Type=TreeView;Name=treMain
```

- ▶ To find the complete object path of the tree view item Bach, beginning with the desktop and ending with the target object itself:

```
Result=SQAGetProperty( "Name=treMain;\ ;ItemText=Bach" ,  
"FullRecognition" , value)
```

Returned value:

```
Type=Window;Name=frmMain;\ ;Type=TreeView;Name=treMain;\ ;  
Type=TVItem;ItemText=Bach
```

- ▶ To find the full-path recognition method for each child object in the currently active window and store it in the array *children()*:

```
Dim children() as String  
Result=SQAGetChildren( ".\" , children)
```

The first three items in the array are:

```
"\ ;Type=Window;Name=frmMain;\ ;Type=Label;Name=lblDollarSign"  
"\ ;Type=Window;Name=frmMain;\ ;Type=Label;Name=lblPrice"  
"\ ;Type=Window;Name=frmMain;\ ;Type=Label;Name=lblPerformer"
```

Object Scripting Status Codes

Object Scripting commands return `sqaSuccess` upon successful execution. If an error occurs, most Object Scripting commands return a status code that identifies the problem. See Appendix C for a listing of the status codes that an Object Scripting command can pass back.

NOTE: If an error occurs during the execution of an Object Scripting command, the command will never log an error message or cause a script to fail. If you want to respond to an error in a particular way, test for the status code and program your response manually.

You can use the `SQABasic Error` function to retrieve a string description of a status code — for example:

```
Result=SQAGetProperty( "Name=myObject" , "Enabled" , value)  
If Result <> sqaSuccess Then  
    SQALogMessage sqaFail, Error$(Result) , "  
End If
```

Managing Custom Verification Points

Robot provides a variety of ways you can verify standard objects — for example, Robot can automatically verify an object's properties (Object Properties verification point), data (Object Data verification point), and text (Alphanumeric verification point).

After script playback, you can view the **baseline data** (captured during recording) in the LogViewer. If the baseline result is different than the **actual data** (captured during playback), you can view both the baseline data and the actual data in a LogViewer Comparator.

You can also verify objects through **custom procedures**, and perform the same kind of verification and LogViewer tasks that Robot performs automatically. For example, if you need to verify the properties of a custom object from build to build, you could write one or more procedures that:

- ▶ Retrieve the target object's properties (similar to recording a standard verification point in Robot).
- ▶ Store the captured data in a .csv file or other file type. This is the baseline file.
- ▶ Play back the custom procedure, and compare the data captured during playback with the data stored in the baseline file (similar to playing back a standard verification point in Robot).
- ▶ Write the baseline data and, if necessary, the actual data, to a log.

The SQABasic verification point management commands can help you with some of these tasks, as described next.

After you capture data for a custom verification point, you can view the data through the LogViewer and the Comparators. The LogViewer and the Comparators display the contents of files of type .csv and .txt. With other file types (such as .doc), the LogViewer opens the appropriate editor to display the file contents.

NOTE: In most cases, the LogViewer and Comparators use the same conventions for .csv files that Microsoft Excel uses. However, the LogViewer and the Comparators ignore leading white space (space or tab characters), where Excel considers leading space characters to be part of the field's value.

Summary of Verification Point Management Commands

To help you perform custom object verification, SQABasic provides a set of verification point management commands.

Most of the verification point management commands return a file name and path — that is, the name and location where LogViewer expects to find a particular data file. For example, if the LogViewer can find a data file, it can display the baseline or actual data in the file.

Here is a summary of the commands:

| Command | Purpose |
|---------------------------------|--|
| SQAVpGetBaselineFileName | Generates the full path and name of a baseline data file. The baseline data file is a copy of the current baseline data file. It is stored in a log for a particular test. |
| SQAVpGetActualFileName | Generates the full path and name of an actual data file captured during playback. |
| SQAVpGetCurrentBaselineFileName | Generates the full path and name of the current baseline data file. You never store the contents of this file in a log directly. Instead, in any given test, you copy the contents of this file to a baseline data file. The latter file is stored in a log for that particular test. |
| SQAVpLog | Writes a custom verification point record to a log. The record is viewable in the LogViewer. |

See Chapter 6, *Command Reference*, for syntax information on these commands.

Current Baseline and Logged Baseline

It is important to understand the difference between the current baseline and the historic, or logged, baseline:

- ▶ The **current baseline** data file contains the data that is currently available for comparison against data captured during playback of a particular test. There is only one current baseline data file per custom verification point.

The current baseline data file can change. For example, suppose a button has the caption OK. In your initial test, you capture this information and store it in the current baseline data file. In subsequent builds, you test to make sure the caption has not changed. But suppose usability testing demonstrates that the caption of this button should be Accept. You change the object's caption, and then you change the current baseline data file so that subsequent tests can be compared against the new caption.

To view the current baseline for a particular verification point, double-click the verification point name in the Robot Asset pane (to left of the script).

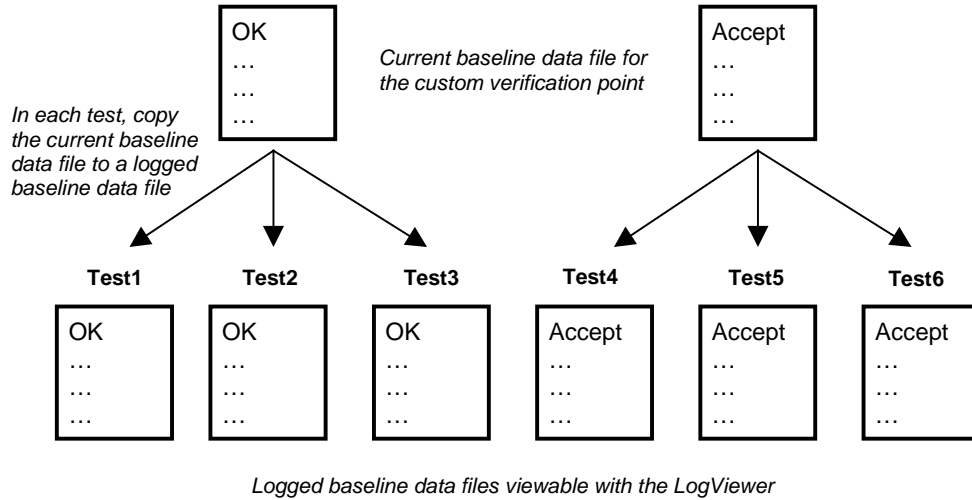
- ▶ The historic, or **logged**, baseline represents the current baseline as it appears in a particular test. The logged baseline data file is copied from the current baseline data file and stored in a log. There is a separate logged baseline data file for each playback result stored in a log. Logs are stored in the repository and can be accessed through a LogViewer Comparator.

To view the logged baseline (and if present, the actual data), double-click the verification point name in the Log Event column of the LogViewer.

You can't change a logged baseline data file through the LogViewer. A logged baseline data file represents the contents of the current baseline data file during a particular test. If you change baseline data through the LogViewer, you are changing the current baseline, not the logged baseline.

You can have many logged baseline data files stored for a given custom verification point. But you can have only one current baseline data file associated with that custom verification point.

In the following figure, the caption OK is changed to Accept between Test3 and Test4. The current baseline data file is modified to accommodate the change:



Actual Data Files

Note that there is not a “current” version of an actual data file. If the actual data captured during playback of a particular test doesn’t match the contents of the current baseline data file:

- ▶ Create an actual data file.
- ▶ Copy the both actual data file and a copy of the current baseline data file to a log. Both files remain part of the log entry for that particular test.

Using the Verification Point Management Commands

This section contains a high-level scenario of typical tasks you perform in custom verification point procedures. It is a guide to help you determine where in your script to use the SQABasic verification point management commands.

All of these tasks are performed in your custom verification point procedures:

1. Create the current baseline data file, as follows:
 - Capture data for the target object. Store the data in a file of any format you choose (such as a .csv file).
 - Call `SQAVpGetCurrentBaselineFileName` to get the file's path and name.

After you call this command, the referenced custom verification point is listed in the Robot Asset pane (to the left of the script) with the script's other verification points. You might have to click View → Refresh to see it.

This step is equivalent to recording a standard verification point in Robot.

2. Play back your first test in Robot. During script playback:
 - Copy the current baseline data file. Assign the new file the name and path returned by `SQAVpGetBaselineFileName`. This action creates a baseline file *for this test only* and stores it in a log.

If the baseline data and the actual data (captured during playback) don't match, you will typically want to keep the baseline data file and the actual data file stored in the log. But even if the baseline data and the actual data do match, you might find it useful to keep a historic record of the baseline data in the log for each test you run.
 - Capture data for the target object (just the way you captured it in step 1). This is the actual data for this test.
 - Compare the contents of the baseline file against the actual data you just captured. If the baseline data matches the actual data (that is, if the custom verification point passes), skip to step 3.
 - If the baseline data does not match the actual data, create a file and write the actual data to it. Store this actual data file in the name and location returned by `SQAVpGetActualFileName`. These actions create an actual data file for this test and store the data file in a log.

The LogViewer can now display the baseline data and the actual data for this test.

3. Call `SQAVpLog` to enter a record into the LogViewer, based on the results of the data comparison in step 2. This record includes the verification point name or a message, and optionally, the notation Pass, Fail, or Warning, in the LogViewer Result column.

4. Repeat steps 2 and 3 for each regression test.

For each test, there is a different copy of the baseline data file saved in a log. If the actual data captured during a test doesn't match the baseline data, an actual data file is also logged for that test.

5. Modify the current baseline data file whenever necessary. There is only one current baseline data file per custom verification point.

You can change the contents of the current baseline data file by changing the baseline data displayed in the log (for example, you can replace the baseline data with the actual data).

Example

The following example illustrates the steps in the previous section:

- ▶ If the script is executed with `runType="GET BASELINE"`, the current baseline data is captured and stored in a file (as described in step 1). This step is similar to recording a standard verification point with Robot.
- ▶ If the script is executed with `runType="GET ACTUAL"`, the captured actual data is compared against the baseline. The baseline data and, on verification point failure, the actual data, are stored in a log for this test run, and the test results are reported in the log (as described in steps 2 and 3). These steps are similar to playing back a standard verification point.

To run this example, copy it from the SQABasic online Help. Run the Help and search for *custom verification points, example of managing* in the Help Index tab.

```
' Script performs the custom verification point MyVP

' Procedure captures data and writes it to the file specified by
' argument DataPath. Returns True if successful, False on error.
Declare Function CustomCaptureData(DataPath As String) As Integer

' Procedure compares the two data files. Returns True if
' successful, False on error.
Declare Function CustomCompareData(BaselinePath As String, _
                                   ActualPath As String) As Integer

Dim runType as String ' Flag retrieval of baseline or actual data
```

Managing Custom Verification Points

```
Sub Main()

Dim currFilepath As String      ' Current baseline data file path
Dim loggedFilepath As String    ' Logged baseline data file path
Dim actFilepath As String       ' Actual data file path
Dim captureResult as Integer    ' Result of MyVP data capture
Dim compareResult as Integer    ' Result of custom verific. pt. MyVP

compareResult = True           ' Default to true.
captureResult = True           ' Default to true.

' ***** TO SIMULATE BASELINE AND ACTUAL DATA CAPTURES: *****
' =====
' * Set runType to "GET BASELINE" to capture baseline data.
' * Set runType to "GET ACTUAL" to capture actual data and
'   to compare it against the baseline.
runType = "GET BASELINE"
'runType = "GET ACTUAL"

If runType = "GET BASELINE" Then

    ' Step 1
    ' =====
    ' This portion captures baseline data. It is similar to
    ' recording a standard verification point.
    ' =====

    ' Get path and file name for current baseline data file
    currFilepath = SQA_Vp_Get_Current_Baseline_File_Name("MyVP","CSV")

    ' Procedure captures data for custom object and records it in
    ' the correct repository location for current baseline files
    captureResult = Custom_Capture_Data(currFilePath)

Else

    ' Step 2
    ' =====
    ' This portion captures the actual data for a particular build
    ' and compares it against the baseline data generated earlier.
    ' Run this portion during playback when testing a build.
    ' =====

    ' Get path and file name for current baseline data file
    currFilepath = SQA_Vp_Get_Current_Baseline_File_Name("MyVP","CSV")

    ' Get LogViewer's path and file name for logged baseline file
    loggedFilepath = SQA_Vp_Get_Baseline_File_Name("MyVP","CSV")

    ' Copy contents of current baseline file to file name/location
    ' where LogViewer expects to find baseline file for this test run
    Call FileCopy(currFilepath,loggedFilepath)

    ' Get LogViewer's path and file name for actual data file
    ' for the data captured in this test run
    actFilepath = SQA_Vp_Get_Actual_File_Name("MyVP","CSV")

End If

End Sub
```

Managing Custom Verification Points

```
' Procedure captures actual data for custom object
captureResult=CustomCaptureData(actFilepath)

' Procedure compares actual data with baseline data
compareResult=CustomCompareData(loggedFilePath,actFilepath)

' Step 3
' Log the results of the custom verification point appropriately
If compareResult = False Then
    Call SQAVpLog(sqaFail,"MyVP","",loggedFilepath,actFilepath)
Else
    Call SQAVpLog(sqaPass,"MyVP","",loggedFilepath,"")
End If

End If

End Sub

' =====
' *** Subroutines ***
' =====

' Call this function to "capture" data and write it to a file
Function CustomCaptureData(DataPath As String) As Integer

    Dim Message As String
    Dim captureType As String

    Open DataPath For Output As #1
    ' Write to baseline or actual data file
    If runType="GET BASELINE" Then
        Write #1,"Baseline data captured during 'recording'"
        captureType = "baseline"
    Else
        Write #1,"Actual data captured during playback"
        captureType = "actual"
    End If
    Close #1

    Message="Now capturing " + captureType + " data." + Chr$(13)
    Message=Message + "Data file path: " + DataPath + Chr$(13)
    MsgBox Message

    CustomCaptureData = True

    If runType="GET BASELINE" Then
        Message="Before you run this example again to simulate "
        Message=Message + "test playback, change" _
            + Chr$(13) + Chr$(13)
        Message=Message + "    runType = ""GET BASELINE"" _
            + Chr$(13) + Chr$(13) + "to" + Chr$(13) + Chr$(13)
        Message=Message + "    runType = ""GET ACTUAL""
        MsgBox Message
    End If

End Function
```

Managing Custom Verification Points

```
' Call this fuction to compare two data files
Function CustomCompareData( BaselinePath As String, _
                           ActualPath As String ) As Integer

    Dim Message As String
    Message="Now comparing baseline and actual data." + Chr$(13)
    Message=Message + "Baseline file path: " + BaselinePath + _
        Chr$(13) + Chr$(13)
    Message=Message + "Actual file path: " + ActualPath
    MsgBox Message

    ' If the function returns True, Pass is reported in the log,
    ' and the actual data file is not stored in the log record
    CustomCompareData = False      ' Baseline/actual data don't match

    Message = "To see the data comparison, double-click "
    Message = Message + ""Fail"" in the LogViewer Result column "
    Message = Message + "for the verification point MyVP."
    MsgBox Message

End Function
```

Ownership of Custom Verification Point Files

Verification point files are associated with, or “owned” by, the following Robot or LogViewer features:

- ▶ LogViewer log – If you delete a log, all of that log’s events (including standard and custom verification point entries) are deleted. This includes the logged baseline and actual data files pointed to by `SQAVpGetBaselineFileName` and `SQAVpGetActualFileName`. However, the current baseline data file pointed to by `SQAVpGetCurrentBaselineFileName` remains.
- ▶ Robot verification point – If you delete a custom verification point from the verification point list in the Asset pane, the associated current baseline data file pointed to by `SQAVpGetCurrentBaselineFileName` is deleted. However, the logged baseline and actual data files remain.
- ▶ Robot script – If you delete a script, all verification points and associated current baseline files are deleted. However, the log associated with the script as well as the logged baseline and actual data files remain.

Comparing Environment States

Robot is shipped with a utility called the VeriTest-Rational Installation Analyzer . This utility is designed to help you detect changes in the environment of a Windows system before and after the performance of some task (such as the installation of an application-under-test) that affects the system's environment.

You can run the Installation Analyzer directly by running ANALYZER.EXE in the default Rational Test 7 directory. Alternatively, you can run the Analyzer within a Robot script, as this section describes.

For more information about the Installation Analyzer, see USING.HTM in the Rational Test 7 directory.

Why Compare Environment States?

Comparing environment states is useful in situations such as these:

- ▶ To test whether a given task in the application-under-test has the *unintended* result of changing the system's environment.
- ▶ To test whether a given task causes *intended* changes in the system's environment, and whether these changes remain consistent in build after build of the application-under-test.

What Environment State Changes Are Detected?

The Installation Analyzer detects environment changes such as:

- ▶ Registry settings
- ▶ Changes to the files WIN.INI, SYSTEM.INI, AUTOEXEC.BAT, and CONFIG.SYS
- ▶ File and file extension changes

Using the Environment State Comparison Commands

You must manually script an environment state comparison.

Comparing Environment States

Use the following SQABasic commands when setting up an environment state comparison:

| Command | Purpose |
|-----------------------------------|--|
| <code>SQAEnvCreateBaseline</code> | Creates a snapshot of a “clean machine” — that is, the state of the environment before one or more tasks are performed that change or are suspected of changing the environment. |
| <code>SQAEnvCreateCurrent</code> | Creates a snapshot of the environment state just after some task is performed that changes or is suspected of changing the environment. |
| <code>SQAEnvCreateDelta</code> | Creates a comparison report of the pre-task and post-task snapshots. Optionally, displays the comparison report in a browser. |

See Chapter 6, *Command Reference*, for syntax information on these commands.

When To Use the Environment State Comparison Commands

Follow these guidelines when using the environment state comparison commands in a script:

- ▶ Call `SQAEnvCreateBaseline` near the beginning of your script, before performing any tasks with your application-under-test that might affect the environment state.
- ▶ After you perform a task with the application-under-test and you want to see if the task has affected the environment state, do the following:
 - Call `SQAEnvCreateCurrent` to capture a snapshot of the current state of the environment.
 - Call `SQAEnvCreateDelta` to compare the current snapshot with the baseline snapshot. Optionally, this command lets you display the comparison results in a browser.

Typically, you call `SQAEnvCreateBaseline` only once in a script. You call `SQAEnvCreateCurrent` and `SQAEnvCreateDelta` whenever you want to test the current state of the environment.

A snapshot captured with `SQAEnvCreateCurrent` can be used as both a post-task snapshot and, at a later point in the script, as a pre-task snapshot. For example, you might want to compare the post-task snapshot captured after you installed the application-under-test with a current snapshot taken after you perform a particular

task with the application-under-test. In this case, both snapshots are created with `SQAEnvCreateCurrent`.

Specifying the Areas of the Environment To Test

By default, the environment state commands take snapshots of the following areas:

- ▶ Your local hard drive
- ▶ The following Registry hives:
 - HKEY_LOCAL_MACHINE
 - HKEY_CURRENT_USER
 - HKEY_CLASSES_ROOT
- ▶ File extensions

Before you run a script that takes a snapshot of the environment, you can change the defaults as follows:

1. Run the Installation Analyzer ANALYZER.EXE. By default, it is located in the Rational Test 7 directory.
2. Click Tools → Options.
3. Specify the areas you want to test.
4. To set the areas you specified as the default areas to test, select Save Settings.
5. Click OK, and then close the Installation Analyzer.

Example of an Environment State Comparison

The following example uses `SQAEnvCreateBaseline` to capture a snapshot of a “clean machine” — that is, a snapshot of the environment before any tasks are performed that might change the state of the environment. The example then calls `SQAEnvCreateCurrent` to capture a snapshot of the environment after each of the following tasks is performed:

- ▶ The sample application Classics Online is installed.
- ▶ The Classics Online application is executed.
- ▶ The Classics Online application is uninstalled.

After each of these tasks, the example calls `SQAEnvCreateDelta` to compare the current state of the environment to the “clean-machine” state captured with `SQAEnvCreateBaseline`. The results are displayed in a browser.

This example requires either Microsoft Systems Installer (MSI) or Windows 2000.

Comparing Environment States

To run this example, copy it from the SQABasic online Help. Run the Help and search for *environment state, complete example* in the Help Index tab.

```
'=====
'
' Script Copyright (c) 2000 by Rational Software
' Author: Pete Jenney - pjenney@rational.com
' Date: 18-Jan-2000
' Notes: Script to demonstrate the use of the SQAEnv* commands in
' the testing process.
' Depends: Microsoft System Installer (MSI) and/or Windows 2000
'=====

'$Include "squtil.sbh"

Sub Main
  Dim Result As Integer
  Dim szSampleInstall As String

  ' Create a baseline snapshot
  Result = SQAEnvCreateBaseLine("CleanMachine")
  If( Result = 0 ) Then
    MsgBox "Failed to create Environment Baseline!",16,"Error!"
    SQAErrorMessage sqaFail,"Failed to create Baseline Snapshot!",""
    Exit Sub
  End If

  ' Install the application
  szSampleInstall = "msiexec /qb+ /i "" & _
    SQAGetDir(SQA_DIR_REPOSITORY) & "Samples\" & _
    "ClassicsOnline.msi""

  StartApplication szSampleInstall

  Result = WindowVP (Exists, _
    "Caption=Rational Test Samples - Classics Online", _
    "VP=Complete Dialog;Wait=1,120")
  If( Result = 0 ) Then
    MsgBox "Completion Dialog never appeared", 16, "Error!"
    SQAErrorMessage sqaFail,"Completion Dialog never appeared",""
    Exit Sub
  End If

  Window SetContext, _
    "Caption=Rational Test Samples - ClassicsOnline;" + _
    "Level=2;State=Disabled", "Activate=0"
  Label Click, "Text=Please wait while Windows configures " + _
    "Rational Test Samples - Classics"
  Window SetContext, _
    "Caption=Rational Test Samples - Classics Online", ""
  PushButton Click, "Text=OK"

  ' Create the PostInstall snapshot
  Result = SQAEnvCreateCurrent("PostInstall")
  If( Result = 0 ) Then
    MsgBox "Failed to create PostInstall Environment Snapshot!", _
      16, "Error!"
    SQAErrorMessage sqaFail, "Failed to create PostInstall " + _
      "Environment Snapshot!", ""
    Exit Sub
  End If
End Sub
```


Comparing Environment States

```
' Create the Delta Report
  Result = SQAEnvCreateDelta("CleanMachine", "PostInstall", 1)
  If( Result = 0 ) Then
    MsgBox "Failed to create CleanMachine/PostInstall Report!", _
      16, "Error!"
    SQALogMessage sqaFail, + _
      "Failed to create CleanMachine/PostInstall Report!",""
  Exit Sub
End If

' Prompt the user to continue or abort
  Result=MsgBox("Press OK to continue or Cancel to halt testing", _
    65, "Action")
  If( Result = 2 ) Then
    SQALogMessage sqaFail, "Testing halted by user at " + _
      "CleanMachine/PostInstall Report", ""
  Exit Sub
End If

' Exercise the application
  CallScript "PlayWithClassics"

' Create the PostRun snapshot
  Result = SQAEnvCreateCurrent("PostRun")
  If( Result = 0 ) Then
    MsgBox "Failed to create PostRun Environment Snapshot!", _
      16, "Error!"
    SQALogMessage sqaFail, _
      "Failed to create PostRun Environment Snapshot!",""
  Exit Sub
End If

' Create the Delta Report
  Result = SQAEnvCreateDelta("PostInstall", "PostRun", 1)
  If( Result = 0 ) Then
    MsgBox "Failed to create PostInstall/PostRun Report!", _
      16, "Error!"
    SQALogMessage sqaFail, _
      "Failed to create PostInstall/PostRun Report!",""
  Exit Sub
End If

' Prompt the user to continue or abort
  Result=MsgBox("Press OK to continue or Cancel to halt testing", _
    65, "Action")
  If( Result = 2 ) Then
    SQALogMessage sqaFail, "Testing halted by user at " + _
      "PostInstall/PostRun Report", ""
  Exit Sub
End If

' Uninstall the application
  szSampleInstall = "msiexec /qb+ /x "" & _
    SQAGetDir(SQA_DIR_REPOSITORY) & "Samples\" & _
    "ClassicsOnline.msi""
  StartApplication szSampleInstall

  Result = WindowVP (Exists, _
    "Caption=Rational Test Samples - Classics Online", _
    "VP=Complete Dialog;Wait=1,120")
```

Displaying Messages in Robot

```
If( Result = 0 ) Then
    MsgBox "Error!", 16, "Completion Dialog never appeared"
    SQALogMessage sqaFail, "Completion Dialog never appeared", ""
    Exit Sub
End If

Window SetContext, _
    "Caption=Rational Test Samples - Classics Online;" + _
    "Level=2;State=Disabled", "Activate=0"
Label Click, "Text=Please wait while Windows configures " + _
    "Rational Test Samples - Classics"
Window SetContext, _
    "Caption=Rational Test Samples - Classics Online", ""
PushButton Click, "Text=OK"

' Create the PostUninstall snapshot
Result = SQAEnvCreateCurrent("PostUninstall")
If( Result = 0 ) Then
    MsgBox "Failed to create PostUninstall Environment " + _
        "Snapshot!", 16, "Error!"
    SQALogMessage sqaFail, "Failed to create PostUninstall " + _
        "Environment Snapshot!", ""
    Exit Sub
End If

' Create the Delta Report
Result = SQAEnvCreateDelta("CleanMachine", "PostUninstall", 1)
If( Result = 0 ) Then
    MsgBox "Failed to create PostUninstall/CleanMachine " + _
        "Report!", 16, "Error!"
    SQALogMessage sqaFail, "Failed to create " + _
        "PostUninstall/CleanMachine Report!", ""
    Exit Sub
End If

End Sub
```

Displaying Messages in Robot

During playback, you can use the following commands to display messages in the Robot console window and in the LogViewer:

| Command | Purpose |
|---------------------|---|
| SQAConsoleWrite | Displays text in the Robot console window. |
| SQAConsoleClear | Remove all text in the Robot console window. |
| SQALogMessdage | Write a message in the LogViewer, and optionally add the notation Pass, Fail, or Warning. |
| SQAScriptCmdFailure | Report a serious runtime error in the LogViewer and stop script playback. |
| SQAVpLog | Write information about custom verification point results to the LogViewer. |

For more information about `SQAVPLog`, see *Managing Custom Verification Points* on page 5-12. The following sections describe the other messaging commands.

NOTE: You can also display a message in a dialog box during runtime with the `SQABasic` command `MsgBox`. However, the dialog box must be explicitly dismissed before the script can continue running.

Displaying Messages in the Console Window

The console window is the area just below the Robot script area. Typically, this area is reserved for your messages.

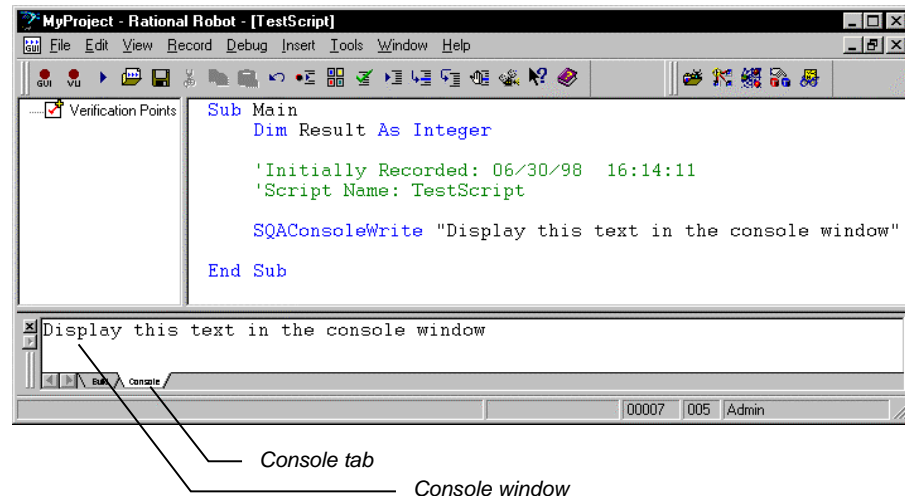
However, Robot may write certain system messages to the console window. For example:

- ▶ Robot reports script command failures in the console window.
- ▶ Robot may display a message in the console window if it detects that you are testing a Java environment that is not ready for Robot due to the use of old class libraries or the lack of a Java enabler.

Displaying the Console Window

If the console window is not displayed, take either or both of these actions to display it:

- ▶ Make sure the Output choice on the View menu is checked.
- ▶ If the Output choice is checked but the console window is still not displayed, click the Console tab in the lower left corner of the Robot main window:



Displaying Messages in Robot

Writing to the Console Window

Use `SQAConsoleWrite` to write text to the console window.

You can insert a carriage return through `Chr$(13)`. For example, to display a blank line between the text `Line1` and `Line2`, call:

```
SQAConsoleWrite "Line1" + Chr$(13) + Chr$(13) + "Line2"
```

If `SQAConsoleWrite` is called multiple times during playback, subsequent messages are appended to the original message.

Removing Messages from the Console Window

Robot clears the console window at the beginning of playback.

To explicitly clear text from the console window, call `SQAConsoleClear`.

Displaying Messages in the LogViewer

You can display messages in the LogViewer through `SQALogMessage` and through `SQAScriptCmdFailure`.

NOTE: `SQAVpLog` also writes messages to the LogViewer. For more information, see *Managing Custom Verification Points* on page 5-12.

Using `SQALogMessage`

This command writes an entry in the Log Event column of the LogViewer. You can use this command to report the success or failure of an event, or to display any informational text you choose.

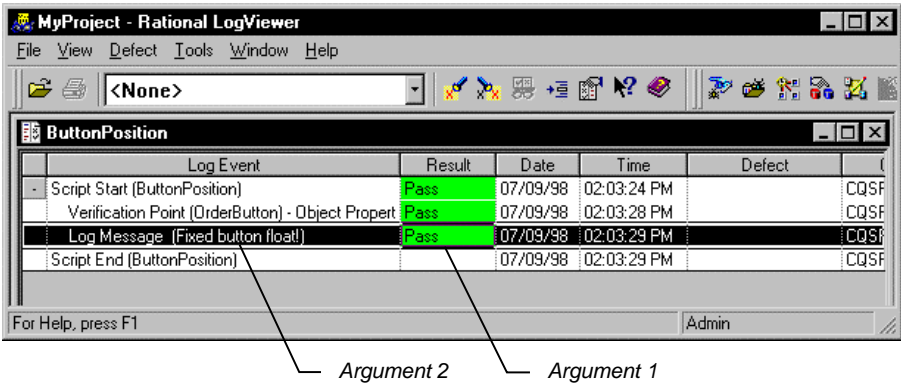
In addition to the information in the Log Event column, you can insert the notation `Pass`, `Fail`, or `Warning` in the Result column. If you insert `Fail`, the LogViewer reports `Fail` for the entire script.

You can also include a description of the event or informational text you display. The description appears in the Result tab of the Log Event Properties dialog box.

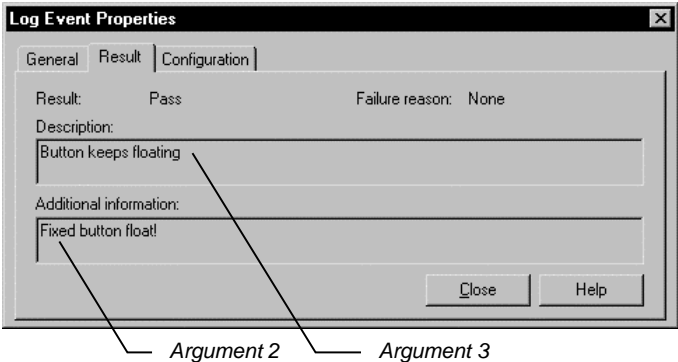
Here is an example of an `SQALogMessage` command and how its arguments are displayed in the LogViewer:

```
SQALogMessage sqaPass, "Fixed button float!", "Button keeps floating"
```

This is the text that is displayed in the LogViewer:



And this is the text that's displayed in the Log Event Properties dialog box:



To display the Result tab of the Log Event Properties dialog box:

1. In the LogViewer, right-click the message you displayed in the Log Event column.
2. Click Properties.
3. Click the Result tab.

Using SQAScriptCmdFailure

This command writes a message to the LogViewer and *stops the execution of the script*. Use this command only for reporting serious events.

SQAScriptCmdFailure takes just one argument — the description of the event.

This command displays the following text in the LogViewer:

- ▶ The text “Script Command Failure” appears in the Log Event column. You can’t modify this text.
- ▶ The notation Fail appears in the Result column. You can’t modify it.
- ▶ The text you provide through this command is displayed in the Result tab of the Log Event Properties dialog box.

In addition, the description you provide of the script command failure and the line where it occurs are displayed in the Robot console window.

Using Datapools

A **datapool** is a test dataset. It supplies data values to the variables in a script during script playback.

Datapools let you automatically pump test data to a script that is being played back repeatedly, allowing the script to send a different set of data to the server in each iteration.

If you do not use a datapool during script playback, the same values (the values that were captured when you recorded the script) are sent to the server each time the script is executed.

For example, suppose you record a script that sends order number 53328 to a database server. If you play back this script 100 times, order number 53328 is sent to the server 100 times. If you use a datapool, each iteration of the script can send a different order number to the server.

You use TestManager to automatically generate datapool data. To access the data in a datapool from a GUI script, you must add the datapool commands manually. For information about creating datapools and manually coding datapool commands for GUI scripts, see the *Using Rational Robot* manual.

Summary of Datapool Commands

These are the SQABasic commands that let you access the data in a datapool.

- ▶ `SQADatapoolClose` – Close the specified datapool.
- ▶ `SQADatapoolFetch` – Move the datapool cursor to the next row.
- ▶ `SQADatapoolOpen` – Open the specified datapool.
- ▶ `SQADatapoolRewind` – Reset the cursor for the specified datapool.
- ▶ `SQADatapoolValue` – Retrieve the value of the specified datapool column.

See Chapter 6, *Command Reference*, for syntax information about these commands.

Accessing External Applications

SQABasic lets you access applications through dynamic data exchange (DDE) and through object linking and embedding (OLE).

Dynamic Data Exchange (DDE)

DDE is a process by which two applications communicate and exchange data. One application can be an SQABasic script.

Opening a DDE Channel

To “talk” to another application and send it data, open a connection (called a DDE channel) using the statement `DDEInitiate`.

`DDEInitiate` requires two arguments:

- ▶ DDE application name. This name is usually the name of the .EXE file used to start the application. Specify the name without the .EXE extension. For example, the DDE name for Microsoft Word is WINWORD.
- ▶ Topic name. This name is usually a filename to get or send data to, although there are some reserved DDE topic names, such as `System`. See the application’s documentation for a list of the available topic names.

The application must already be running before you can open a DDE channel. To start an application, use the `Shell` command.

Communicating with the Application

After you open a channel to an application, you can get text and numbers (`DDERequest`), send text and numbers (`DDEPoke`), or send commands (`DDEExecute`). See the application’s documentation for a list of supported DDE commands.

To make sure the application performs a DDE task as expected, use `DDEAppReturnCode`. If an error does occur, your program can notify the user.

Closing the Channel

When you’re finished communicating with the application, you should close the DDE channel using `DDETerminate`. Because you have a limited number of channels available at once (depending on the operating system in use and the amount of memory you have available), it’s a good idea to close a channel as soon as you finish using it.

Objects

SQABasic supports OLE2 Object Handling. OLE2 provides the ability to link and embed objects from one application into another. Key OLE2 terms:

- ▶ **Objects** are the end products of a software application, such as a spreadsheet, graph, or document objects, and OLE Automation objects. Each application has its own set of properties and methods that change the characteristics of an object.
- ▶ **Properties** affect how an object behaves. For example, width is a property of a range of cells in a spreadsheet, colors are a property of graphs, and margins are a property of word processing documents.
- ▶ **Methods** cause the application to do something to an object. Examples are Calculate for a spreadsheet, Snap to Grid for a graph, and AutoSave for a document.

SQABasic lets you access an external object and use the originating application to change properties and methods of that object.

Before you can use an object in a procedure, you must access the application associated with the object by assigning the object to an object variable. Then you attach an object name (with or without properties and methods) to the variable to manipulate the object.

For example code, see the Overview topic for the `Set` statement in the SQABasic online Help.

Step 1: Create an Object Variable to Access the Application

In the lines of code below, the `Dim` statement creates an object variable called `visio`. The `Set` statement associates the variable `visio` with the VISIO application by calling the `GetObject` function:

```
Dim visio as Object
...
Set visio = GetObject(,"visio.application") ' find Visio
```

Note that `GetObject` is used if the application is already open on the Windows desktop. Use `CreateObject` if the application is not open.

Step 2: Use Methods and Properties to Act on Objects

To access an object, property or method, use this syntax:

```
appvariable.object
appvariable.object.property
appvariable.object.method
```

For example, `visio.documents.count` references the Count method of the Document object for the VISIO application.

Optionally, you can create a second object variable and assign the Document object to it using VISIO's Document method, as the following Set statement shows:

```
dim doc as Object
dim I as Integer, doccount as Integer
dim msgtext as String
...
doccount = visio.documents.count
If doccount = 0 then
    MsgBox "No open Visio documents."
else
    msgtext = "The open files are: " & Chr$(13)
    For i = 1 to doccount
        Set doc = visio.documents(i)
        msgtext = msgtext & chr$(13) & doc.name
    Next I
End If
```

NOTE: Object, property, and method names vary from one application to another. See the application's documentation for the applicable names to use.

Accessing External Applications

▶ ▶ ▶ Part III

Command Reference

▶ ▶ ▶ C H A P T E R 6

Command Reference

This command reference contains the following categories of information:

- ▶ The Microsoft Basic **functions**, **statements**, and **operators** that SQABasic supports.
- ▶ SQABasic command additions to standard Basic. Most additions fall into these categories:

Datapool commands – Access data in a datapool.

Object Scripting commands – Access objects and object properties.

Timing and Coordination commands – Time user activities and control the rate of script playback.

User Action commands – Capture a user's keyboard and mouse actions during recording.

Utility commands – Perform a variety of tasks in an SQABasic script.

Verification Point commands – Compare the results of a user action during recording to the results of the same action when it's later played back.

In addition to the above categories of command additions, SQABasic provides the following new commands — the `Assert` statement, the `GetField` function, the `SetField` function, and the metacommands `'$CStrings`, `'$Include`, and `'NoCStrings`.

▶▶▶ SQA ▶

NOTE: The icon in the margin appears next to the names of SQABasic command additions. You may find this icon useful when scanning for the additions.

Abs

Abs

Function

Description Returns the absolute value of a number.

Syntax **Abs** (*number*)

| Syntax Element | Description |
|----------------|-------------------------------|
| <i>number</i> | Any valid numeric expression. |

Comments The data type of the return value matches the type of the *number*. If *number* is a Variant string (VarType 8), the return value will be converted to VarType 5 (Double). If the absolute value evaluates to VarType 0 (Empty), the return value will be VarType 3 (Long).

Example This example finds the difference between two variables, `oldacct` and `newacct`.

```
Sub main
Dim oldacct, newacct, count
oldacct=InputBox("Enter the oldacct number")
newacct=InputBox("Enter the newacct number")
count=Abs(oldacct-newacct)
MsgBox "The absolute value is: " &count
End Sub
```

See Also

| | |
|-----|---------|
| Exp | Rnd |
| Fix | Sgn |
| Int | Sqr |
| Log | Variant |

AnimateControl

User Action Command

»»SQA»

Description Performs an action on an animation control.

Syntax **AnimateControl** *action%*, *recMethod\$*, *parameters\$*

| Syntax Element | Description |
|---------------------|---|
| <i>action%</i> | <p>One of these mouse actions:</p> <ul style="list-style-type: none"> ▶ <i>MouseClicked</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> must contain <i>Coords=x,y</i>. ▶ <i>MouseDown</i>. The dragging of the mouse while mouse buttons and/or shifting keys (Ctrl, Alt, Shift) are pressed. When <i>action%</i> contains a mouse-drag value, <i>parameters\$</i> must contain <i>Coords=x1,y1,x2,y2</i>. <p>See Appendix E for a list of mouse click and drag values.</p> |
| <i>recMethod\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <i>ID=%</i>. The object's internal Windows ID. ▶ <i>Name=\$</i>. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be <i>Command1</i>. ▶ <i>ObjectIndex=%</i>. The number of the object among all objects of the same type in the same window. ▶ <i>State=\$</i>. An optional qualifier for any other recognition method. There are two possible values for this setting: <i>Enabled</i> and <i>Disabled</i>. The default state is the state of the current context window (as set in the most recent <i>Window SetContext</i> command), or <i>Enabled</i> if the state has not been otherwise declared. ▶ <i>Text=\$</i>. The text displayed on the object. ▶ <i>VisualText=\$</i>. An optional setting used to identify an object by its visible text. It is for user clarification only and does not affect object recognition. |
| <i>parameters\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <i>Coords=x,y</i>. If <i>action%</i> is a mouse click, specifies the coordinates of the click, relative to the top left of the object. ▶ <i>Coords=x1,y1,x2,y2</i>. If <i>action%</i> is a mouse drag, specifies the coordinates, where <i>x1,y1</i> are the starting coordinates of the drag, and <i>x2,y2</i> are the ending coordinates. The coordinates are relative to the top left of the object. |

Comments None.

AnimateControlVP

Example This example clicks the first animation control in the window (ObjectIndex=1) at x,y coordinates of 50,25.

```
AnimateControl Click, "ObjectIndex=1", "Coords=50,25"
```

See Also AnimateControlVP

AnimateControlVP

Verification Point Command



Description Establishes a verification point for an animation control.

Syntax *Result* = **AnimateControlVP**(*action%*, *recMethod\$*, *parameters\$*)

| Syntax Element | Description |
|--------------------|--|
| <i>action%</i> | <p>The type of verification to perform. Valid values:</p> <ul style="list-style-type: none">▶ CompareNumeric. Captures the numeric value of the text of the object and compares it to the value of <i>parameters\$ Value or Range</i>. <i>parameters\$ VP</i> and either <i>Value or Range</i> are required; <i>ExpectedResult</i> and <i>Wait</i> are optional.▶ CompareProperties. Captures object properties information for the object and compares it to a recorded baseline. <i>parameters\$ VP</i> is required; <i>ExpectedResult</i> and <i>Wait</i> are optional.▶ CompareText. Captures the text of the object and compares it to a recorded baseline. <i>parameters\$ VP</i> and <i>Type</i> are required; <i>ExpectedResult</i> and <i>Wait</i> are optional. |
| <i>recMethod\$</i> | <p>Valid values:</p> <ul style="list-style-type: none">▶ ID=%. The object's internal Windows ID.▶ Name=\$. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be Command1.▶ ObjectIndex=%. The number of the object among all objects of the same type in the same window.▶ Text=\$. The text displayed on the object. |





| Syntax Element | Description |
|---------------------------|---|
| <code>parameters\$</code> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <code>ExpectedResult=%</code>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> - <code>PASS</code>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail. - <code>FAIL</code>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail. ▶ <code>Range=&, &</code>. Used with the action <code>CompareNumeric</code> when a numeric range comparison is being performed, as in <code>Range=2, 12</code> (test for numbers in this range). The values are inclusive. ▶ <code>Type=\$</code>. Specifies the verification method to use for <code>CompareText</code> actions. The possible values are: <code>CaseSensitive</code>, <code>CaseInsensitive</code>, <code>FindSubStr</code>, <code>FindSubStrI</code> (case insensitive), and <code>UserDefined</code>. See <i>Comments</i> for more information. If <code>UserDefined</code> is specified, two additional parameters are required: <ul style="list-style-type: none"> - <code>DLL=\$</code>. The full path and file name of the library that contains the function - <code>Function=\$</code>. The name of the custom function to use in comparing the text ▶ <code>Value=&</code>. Used with the action <code>CompareNumeric</code> when a numeric equivalence comparison is being performed, as in <code>Value=25</code> (test against the value 25). ▶ <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points. ▶ <code>Wait=%, %</code>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10, 40</code> (retry the test every 10 seconds, but time out the test after 40 seconds). |

Comments This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

With the `Type=$` parameter, `CaseSensitive` and `CaseInsensitive` require a full match between the current baseline text and the text captured during playback. With `FindSubStr` and `FindSubStrI`, the current baseline can be a

AppActivate

substring of the text captured during playback. The substring can appear anywhere in the playback text. To modify the current baseline text, double-click the verification point name in the Robot Asset pane (to the left of the script).

Example This example captures the properties of the first animation control in the window (ObjectIndex=1) and compares them to the recorded baseline in verification point TEST1A.

```
Result = AnimateControlVP (CompareProperties, "ObjectIndex=1",  
"VP=TEST1A")
```

See Also AnimateControl

AppActivate

Statement

Description Activates an application window.

Syntax **AppActivate** *title*\$

| Syntax Element | Description |
|-----------------|---|
| <i>title</i> \$ | A string expression for the title-bar name of the application window to activate. |

Comments *Title* must match the name of the window character for character, but the comparison is not case-sensitive. For example, "Notepad" is the same as "notepad" or "NOTEPAD". If there is more than one window with a name matching *title*, a window is chosen at random.

AppActivate changes the focus to the specified window but does not change whether the window is minimized or maximized. Use *AppActivate* with the *InputKeys* statement to send keys to another application.

Example This example runs Microsoft Notepad and types some text into the editor.

```
Sub Main  
  StartApplication("notepad.exe")  
  AppActivate "Untitled - Notepad"  
  DoEvents  
  InputKeys "Hello, world.{ENTER}"  
End Sub
```

See Also *InputKeys*
Shell

Asc

Function

Description Returns an integer corresponding to the character code of the first character in the specified string.

Syntax **Asc**(*string\$*)

| Syntax Element | Description |
|-----------------|--|
| <i>string\$</i> | A string expression of one or more characters. |

Comments To change a character code to a character string, use `Chr`.
To obtain the first byte of a string, use `AscB`.

Example This example asks the user for a letter and returns its ASCII value.

```
Sub main
    Dim userchar
    userchar=InputBox("Type a letter:")
    MsgBox "The ASC value for " & userchar & " is: " & Asc(userchar)
End Sub
```

See Also `Chr`

Assert

Statement

»»»SQA»

Description Triggers a runtime error if the condition specified is `FALSE`.

Syntax **Assert** *condition*

| Syntax Element | Description |
|------------------|---|
| <i>condition</i> | A numeric or string expression that can evaluate to <code>TRUE</code> or <code>FALSE</code> . |

Comments The `Assert` statement should be used to handle an application-specific error. An assertion error cannot be trapped by the `On Error` statement.

Use the `Assert` statement to ensure that a script is performing as expected.

Example None.

See Also None.

Command Reference

Atn

Atn

Function

Description Returns the angle (in radians) for the arc tangent of the specified number.

Syntax **Atn**(*number*)

| Syntax Element | Description |
|----------------|-------------------------------|
| <i>number</i> | Any valid numeric expression. |

Comments The **Atn** function assumes *number* is the ratio of two sides of a right triangle: the side opposite the angle to find and the side adjacent to the angle. The function returns a single-precision value for a ratio expressed as an integer, a currency, or a single-precision numeric expression. The return value is a double-precision value for a long, Variant or double-precision numeric expression.

To convert radians to degrees, multiply by (180/PI). The value of PI is approximately 3.14159.

Example This example finds the roof angle necessary for a house with an attic ceiling of 8 feet (at the roof peak) and a 16 foot span from the outside wall to the center of the house.

```
Sub main
  Dim height, span, angle, PI
  PI=3.14159
  height=8
  span=16
  angle=Atn(height/span)*(180/PI)
  MsgBox "The angle is " & Format(angle, "##.##") & " degrees"
End Sub
```

See Also

Cos
Sin
Tan
Derived Trigonometric functions (Appendix D)

Beep

Statement

Description Produces a tone through the computer speaker.

Syntax **Beep**

Comments The frequency and duration of the tone depends on the hardware.

Example This example beeps and displays a message in a box if the variable *balance* is less than 0. (If you have a set of speakers hooked up to your computer, you might need to turn them on to hear the beep.)

```
Sub main
  Dim expenses, balance, msgtext
  balance=InputBox("Enter your account balance")
  expenses=1000
  balance=balance-expenses
  If balance<0 then
    Beep
    MsgBox "I'm sorry, your account is overdrawn."
  Else
    MsgBox "Your balance minus expenses is: " & balance
  End If
End Sub
```

See Also InputBox Print
MsgBox

Begin Dialog...End Dialog

Statement

Description Begins and ends a definition of a dialog box record.

Syntax **Begin Dialog** *dialogName* [*x*, *y*,] *dx*, *dy* [, *caption\$*]
 [, *.dialogfunction*]
 ... ' dialog box definition statements
End Dialog

| Syntax Element | Description |
|------------------------|--|
| <i>dialogName</i> | The record name for the dialog box definition. |
| <i>x</i> , <i>y</i> | The coordinates for the upper left corner of the dialog box. |
| <i>dx</i> , <i>dy</i> | The width and height of the dialog box (relative to <i>x</i> and <i>y</i>). |
| <i>caption\$</i> | The title for the dialog box. |
| <i>.dialogfunction</i> | A function to process user actions in the dialog box. |

Begin Dialog...End Dialog

Comments To create and display a dialog box:

1. Define the dialog box and its controls using the `Begin Dialog...End Dialog` statements and the object definition statements (such as `TextBox`, `OKButton`).
2. Optionally, use the `.dialogfunction` argument to call a function you define to handle user actions in the dialog box.
3. Use the `Dim` statement to declare an instance of the dialog box you defined in step 1.
4. Display the dialog box using either the `Dialog` function or the `Dialog` statement.

For example code, see the Overview topic for the `Begin Dialog...End Dialog` statement in the SQABasic online Help.

The x and y coordinates are relative to the upper left corner of the client area of the parent window. The x argument is measured in units that are 1/4 the average width of the system font. The y argument is measured in units 1/8 the height of the system font. For example, to position a dialog box 20 characters in, and 15 characters down from the upper left hand corner, enter 80, 120 as the x , y coordinates. If these arguments are omitted, the dialog box is centered in the client area of the parent window.

The dx argument is measured in 1/4 system-font character-width units. The dy argument is measured in 1/8 system-font character-width units. For example, to create a dialog box 80 characters wide, and 15 characters in height, enter 320, 120 for the dx , dy coordinates.

If the `caption$` argument is omitted, a standard default caption is used.

The optional `.dialogfunction` function must be defined (using the `Function` statement) or declared (using `Dim`) before being used in the `Begin Dialog` statement. Define the `dialogfunction` with the following three arguments:

```
Function dialogfunction% (id$, action%, supvalue&)  
... 'function body  
End Function
```

Here are the descriptions of the arguments:

| Argument | Description |
|-----------------------|---|
| <i>id\$</i> | <p>The text string that identifies the dialog control that triggered the call to the dialog function (usually because the user changed this control).</p> <p><i>id\$</i> is the same value for the dialog control that you use in the definition of that control. For example, the <i>id\$</i> value for a text box is <code>Text1</code> if it is defined this way:</p> <pre>Textbox 271, 78, 33, 18, .Text1</pre> <p><i>id\$</i> values are case-sensitive and don't include the dot (.) that appears before the ID in the definition of the control.</p> |
| <i>action%</i> | <p>One of the following values. The values identify the reason why the dialog function was called.</p> <ol style="list-style-type: none"> 1. Dialog box initialization. This value is passed before the dialog box becomes visible. 2. Command button selected or dialog box control changed (except typing in a text box or combo box). 3. Change in a text box or combo box. This value is passed when the control loses the input focus: the user presses the TAB key or clicks another control. 4. Change of control focus. <i>Id\$</i> is the id of the dialog control gaining focus. <i>Suppvalue&</i> contains the numeric id of the control losing focus. A dialog function cannot display a message box or dialog box in response to an action value 4. 5. An idle state. As soon as the dialog box is initialized (<i>action%</i> = 1), the dialog function will be continuously called with <i>action%</i> = 5 if no other action occurs. If <i>dialog function</i> wants to receive this message continuously while the dialog box is idle, return a non-zero value. If 0 (zero) is returned, <i>action%</i> = 5 will be passed only while the user is moving the mouse. For this action, <i>Id\$</i> is equal to empty string (" ") and <i>suppvalue&</i> is equal to the number of times action 5 was passed before. |
| <i>Suppvalue&</i> | <p>Gives more specific information about why the dialog function was called. If the user clicks a command button or changes a dialog box control, <i>action%</i> returns 2 or 3 and <i>suppvalue&</i> identifies the control affected. The value returned depends on the type of control or button the user changed or clicked. See the table below for valid <i>Suppvalue&</i> values.</p> |

Begin Dialog...End Dialog

The following table summarizes the possible values for *suppvalue&*:

| Value | Control |
|--|--|
| Any number | List box. Number of the item selected, 0-based. |
| 1 - selected 0 - cleared -1 - grayed | Check box. |
| Any number | Option button. Number of the option button in the option group, 0-based. |
| Any number | Text box. Number of characters in the text box. |
| Any number | Combo box. The number of the item selected (0-based) when <i>action%=2</i> , or the number of characters in its text box when <i>action%=3</i> . |
| 1 | OK button. |
| 2 | Cancel button. |

In most cases, the return value of *dialogfunction* is ignored. The exceptions are a return value of 2 or 5 for *action%*. If the user clicks the OK button, Cancel button, or a command button (as indicated by an *action%* return value of 2 and the corresponding *id\$* for the button clicked), and the dialog function returns a non-zero value, the dialog box will *not* be closed.

Unless the `Begin Dialog` statement is followed by at least one other dialog-box definition statement and the `End Dialog` statement, an error will result. The definition statements must include an `OKButton`, `CancelButton` or `Button` statement. If this statement is left out, there will be no way to close the dialog box, and the script will be unable to continue executing.

Example

This example defines and displays a dialog box with each type of item in it: list box, combo box, buttons, etc.

```
Sub main
  Dim ComboBox1() as String
  Dim ListBox1() as String
  Dim DropDownList1() as String
  Dim x as Integer
  ReDim ListBox1(0)
  ReDim ComboBox1(0)
  ReDim DropDownList1(3)
  ListBox1(0)="C:\\"
  ComboBox1(0)=Dir("C:\*.*)"
  For x=0 to 2
    DropDownList1(x)=Chr(65+x) & ":"
  Next x
```



```

Begin Dialog UserDialog 274, 171, "SQABasic Dialog Box"
  ButtonGroup .ButtonGroup1
  Text 9, 3, 69, 13, "Filename:", .Text1
  DropComboBox 9, 14, 81, 119, ComboBox1(), .ComboBox1
  Text 106, 2, 34, 9, "Directory:", .Text2
  ListBox 106, 12, 83, 39, ListBox1(), .ListBox2
  Text 106, 52, 42, 8, "Drive:", .Text3
  DropListBox 106, 64, 95, 44, DropListBox1(), .DropListBox1
  CheckBox 9, 142, 62, 14, "List .TXT files", .CheckBox1
  GroupBox 106, 111, 97, 57, "File Range"
  OptionGroup .OptionGroup2
    OptionButton 117, 119, 46, 12, "All pages", .OptionButton3
    OptionButton 117, 135, 67, 8, "Range of pages", .OptionButton4
  Text 123, 146, 20, 10, "From:", .Text6
  Text 161, 146, 14, 9, "To:", .Text7
  TextBox 177, 146, 13, 12, .TextBox4
  TextBox 145, 146, 12, 11, .TextBox5
  OKButton 213, 6, 54, 14
  CancelButton 214, 26, 54, 14
  PushButton 213, 52, 54, 14, "Help", .Push1
End Dialog
Dim mydialog as UserDialog
On Error Resume Next
Dialog mydialog
If Err=102 then
  MsgBox "Dialog box canceled."
End If
End Sub

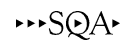
```

See Also

| | | |
|--------------|--------------|----------------|
| Button | Dialog | OptionGroup |
| ButtonGroup | DropComboBox | Picture |
| CancelButton | GroupBox | StaticComboBox |
| Caption | ListBox | Text |
| CheckBox | OKButton | TextBox |
| ComboBox | OptionButton | |

Browser

Utility Command



Description Performs an action on a Web browser.

Syntax **Browser** *action\$, recMethod\$, parameters\$*

| Syntax Element | Description |
|-----------------|--|
| <i>action\$</i> | The following actions: <ul style="list-style-type: none"> ▶ Back. Navigate back one page in the history list. The equivalent of clicking the Back button on the browser toolbar. ▶ Forward. Navigate forward one page in the history list. The equivalent of clicking the Forward button on the browser toolbar. |



▶ ▶ ▶

| Syntax Element | Description |
|--------------------|---|
| | <ul style="list-style-type: none"> ▶ <code>GoToURL</code>. Navigate to the specified URL. Specify the URL ID string as the <i>recMethod\$</i>. ▶ <code>NewPage</code>. Robot waits for a new Web page to load before continuing with the next script command. Robot records a <code>NewPage</code> action just before the first action or verification point recorded on the current page. You can use the optional <i>parameter\$ Wait</i> with this action. ▶ <code>Refresh</code>. Reload the current page in the browser. The equivalent of clicking the Refresh button on the browser toolbar. ▶ <code>SetApplet</code>. Indicates that the <code>BROWSER</code> command is specifying the parent Java object for subsequent Java commands. The parent object is identified in <i>recMethod\$</i>. ▶ <code>SetFrame</code>. Specifies the Web page frame for subsequent script commands. Requires the <i>recMethod\$ Name</i>. ▶ <code>StopLoading</code>. Stop loading the current page in the browser. The equivalent of clicking the Stop button on the browser toolbar. ▶ <code>CloseWin</code>. Close the browser. |
| <i>recMethod\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <code>[empty quotes]</code>. Robot waits for the page to load in the top-most frame. ▶ <code>HTMLId=\$</code>. The text from the ID attribute of the HTML object. ▶ <code>HTMLTitle=\$</code>. The text from the Title attribute of the HTML object. Used with <i>action\$ NewPage</i>. If the document has no title, <code>HTMLTitle</code> is blank. If <i>recMethod\$</i> identifies a document through a frame (with the <code>Type</code> qualifier) but no title, Robot assumes that the next documented loaded is the intended new page. ▶ <code>Index=%</code>. The number of the object among all objects identified with the same base recognition method. Typically, <code>Index</code> is used after another recognition method qualifier — for example, <code>Name=\$; Index=%</code>. |

▶ ▶ ▶



| Syntax Element | Description |
|---------------------------|--|
| | <ul style="list-style-type: none"> ▶ <code>JavaCaption=\$</code>. The text of the Java window caption. The caption can be used to identify the parent Java object when the object has no programmatic name. The wildcards <code>?</code> and <code>*</code> are supported. (See <i>Establishing Context through a Window Command</i> in Chapter 4 for information.) Used only with window-based parent objects, not with browser-based applets. ▶ <code>JavaClass=\$</code>. The Java class name. The class name can be used to identify the parent Java object when the object has no programmatic name or window caption. ▶ <code>Name=\$</code>. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for an applet might be Color Chooser. ▶ <code>Type=\$</code>. Used to identify the specific frame within a Frameset. |
| <code>parameters\$</code> | <p>Valid value:</p> <ul style="list-style-type: none"> ▶ <code>Wait=%</code>. An optional identifier that specifies the number of seconds that Robot will wait for the page specified in <code>recMethod\$</code> to load (or, if no page is specified, for the next page to load). If a page is specified in <code>recMethod\$</code>, the <code>Wait</code> value applies to that page only, not to any pages that may load between the time the <code>Browse</code> command is executed and the loading of the specified page. If the page does not load within the specified time, Robot issues a warning, and the script continues executing. If you do not specify a <code>Wait</code> time, Robot waits 30 seconds. Wait applies only to <code>action\$NewPage</code>. |

Comments Before using this command, use `StartBrowser` to run the browser and enable Web object recognition.

You can also enable Web object recognition by opening the Web page `rbtstart.htm`. This web page references the Rational ActiveX Test Control, which enables object recognition in subsequent activity within the browser. By default, `rbtstart.htm` is located in:

`C:\Program Files\Rational\Rational Test 7`

Button

Once you enable Web object recognition in Robot, Web object recognition is enabled for all subsequent actions against that browser and any new browser windows opened from that browser. For example, if you run `StartBrowser` to open `Browser1`, and then from `Browser1` you open `Browser2` through a JavaScript command or by holding down the Shift key and clicking on a link in Internet Explorer, Web testing is enabled for both `Browser1` and `Browser2`.

If a timeout occurs during a `NewPage` action, Robot returns a warning.

`Browser` can be used to specify the parent Java object for subsequent user action and verification point commands that act upon child objects in the Java environment. However, `Browser` cannot be used in this way with Object Scripting commands. For more information about parent and child Java objects, see *Recognition Methods in Java Commands* in Chapter 4.

Example This example waits for a new Web page to load before executing the next script command.

```
Browser NewPage, "HTMLTitle=My Web Page", ""
```

See Also `StartBrowser`

Button Statement

Description Defines a custom push button.

Syntax A `Button x, y, dx, dy, text$ [, .id]`

Syntax B `PushButton x, y, dx, dy, text$ [, .id]`

| Syntax Element | Description |
|---------------------|--|
| <code>x, y</code> | The position of the button relative to the upper left corner of the dialog box. |
| <code>dx, dy</code> | The width and height of the button. |
| <code>text\$</code> | The name for the push button. If the width of this string is greater than <code>dx</code> , trailing characters are truncated. |
| <code>.id</code> | An optional identifier used by the dialog statements that act on this control. |

Comments A *dy* value of 14 typically accommodates text in the system font.

Use this statement to create buttons other than OK and Cancel. Use this statement in conjunction with the `ButtonGroup` statement. The two forms of the statement (`Button` and `PushButton`) are equivalent.

Use the `Button` statement only between a `Begin Dialog` and an `End Dialog` statement.

Example This example defines a dialog box with a combination list box and three buttons.

```

Sub main
  Dim fchoices as String
  fchoices="File1" & Chr(9) & "File2" & Chr(9) & "File3"
  Begin Dialog UserDialog 185, 94, "SQABasic Dialog Box"
    Text 9, 5, 69, 10, "Filename:", .Text1
    DropComboBox 9, 17, 88, 71, fchoices, .ComboBox1
    ButtonGroup .ButtonGroup1
    OKButton 113, 14, 54, 13
    CancelButton 113, 33, 54, 13
    Button 113, 57, 54, 13, "Help", .Push1
  End Dialog
  Dim mydialog as UserDialog
  On Error Resume Next
  Dialog mydialog
  If Err=102 then
    MsgBox "Dialog box canceled."
  End If
End Sub

```

See Also

| | | |
|---------------------------|---------------------------|-----------------------------|
| <code>Begin Dialog</code> | <code>ComboBox</code> | <code>OptionButton</code> |
| <code>End Dialog</code> | <code>DropComboBox</code> | <code>OptionGroup</code> |
| <code>ButtonGroup</code> | <code>DropListBox</code> | <code>Picture</code> |
| <code>CancelButton</code> | <code>GroupBox</code> | <code>StaticComboBox</code> |
| <code>Caption</code> | <code>ListBox</code> | <code>Text</code> |
| <code>CheckBox</code> | <code>OKButton</code> | <code>TextBox</code> |

ButtonGroup Statement

Description Begins the definition of a group of custom buttons for a dialog box.

Syntax **ButtonGroup** *.field*

| Syntax Element | Description |
|----------------|--|
| <i>.field</i> | The field to contain the user's custom button selection. |

Calendar

Comments If `ButtonGroup` is used, it must appear before any `PushButton` (or `Button`) statement that creates a custom button (one other than OK or Cancel). Only one `ButtonGroup` statement is allowed within a dialog box definition.

Use the `ButtonGroup` statement only between a `Begin Dialog` and an `End Dialog` statement.

Example This example defines a dialog box with a group of three buttons.

```
Sub main
  Begin Dialog UserDialog 34,0,231,140, "SQABasic Dialog Box"
    ButtonGroup .bg
    PushButton 71,17,88,17, "&Button 0"
    PushButton 71,50,88,17, "&Button 1"
    PushButton 71,83,88,17, "&Button 2"
  End Dialog
  Dim mydialog as UserDialog
  Dialog mydialog
  MsgBox "Button " & mydialog.bg & " was pressed."
End Sub
```

See Also

| | | |
|---------------------------|---------------------------|-----------------------------|
| <code>Begin Dialog</code> | <code>ComboBox</code> | <code>OptionButton</code> |
| <code>End Dialog</code> | <code>DropComboBox</code> | <code>OptionGroup</code> |
| <code>Button</code> | <code>DropListBox</code> | <code>Picture</code> |
| <code>CancelButton</code> | <code>GroupBox</code> | <code>StaticComboBox</code> |
| <code>Caption</code> | <code>ListBox</code> | <code>Text</code> |
| <code>CheckBox</code> | <code>OKButton</code> | <code>TextBox</code> |

Calendar

User Action Command



Description Performs an action on a month calendar control.

Syntax `Calendar` *action%*, *recMethod\$*, *parameters\$*

| Syntax Element | Description |
|----------------|---|
| <i>action%</i> | One of these mouse actions: <ul style="list-style-type: none">▶ <i>MouseClicked</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> must contain <code>Coords=x,y</code>.▶ <i>MouseDown</i>. The dragging of the mouse while mouse buttons and/or shifting keys (Ctrl, Alt, Shift) are pressed. When <i>action%</i> contains a mouse-drag value, <i>parameters\$</i> must contain <code>Coords=x1,y1,x2,y2</code>. See Appendix E for a list of mouse click and drag values. |





| Syntax Element | Description |
|----------------------|---|
| <i>recMethod</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ ID=% . The object's internal Windows ID. ▶ Label=\$. The text of the label object that immediately precedes the control in the internal order (Z order) of windows. ▶ Name=\$. A unique name that a developer assigns to an object to identify the object in the development environment. For example, the object name for a command button might be Command1. ▶ ObjectIndex=% . The number of the object among all objects of the same type in the same window. ▶ Text=\$. The text displayed on the object. ▶ VisualText=\$. An optional setting used to identify an object by its prior label. It is for user clarification only and does not affect object recognition. |
| <i>parameters</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ Coords=<i>x,y</i> . If <i>action%</i> is a mouse click, specifies the coordinates of the click, relative to the top left of the object. ▶ Coords=<i>x1,y1,x2,y2</i> . If <i>action%</i> is a mouse drag, specifies the coordinates, where <i>x1,y1</i> are the starting coordinates of the drag, and <i>x2,y2</i> are the ending coordinates. The coordinates are relative to the top left of the object. |

Comments None.

Example This example clicks the month calendar control labeled "Select a Date" at *x,y* coordinates of 135,105.

```
Calendar Click, "Label=Select a Date", "Coords=135,105"
```

See Also CalendarVP
DateTime

CalendarVP

Verification Point Command

»»SQA»

Description Establishes a verification point for a month calendar control.

Syntax `Result = CalendarVP (action%, recMethod$, parameters$)`

| Syntax Element | Description |
|---------------------------|---|
| <code>action%</code> | <p>The type of verification to perform. Valid values:</p> <ul style="list-style-type: none"> ▶ <code>CompareProperties</code>. Captures object properties information for the object and compares it to a recorded baseline. <code>parameters\$ VP</code> is required; <code>ExpectedResult</code> and <code>Wait</code> are optional. |
| <code>recMethod\$</code> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <code>ID=%</code>. The object's internal Windows ID. ▶ <code>Label=\$</code>. The text of the label object that immediately precedes the control in the Z order of windows. ▶ <code>Name=\$</code>. A unique name that a developer assigns to an object to identify the object in the development environment. For example, the object name for a command button might be <code>Command1</code>. ▶ <code>ObjectIndex=%</code>. The number of the object among all objects of the same type in the same window. ▶ <code>Text=\$</code>. The text displayed on the object. ▶ <code>VisualText=\$</code>. An optional setting used to identify an object by its prior label. It is for user clarification only and does not affect object recognition. |
| <code>parameters\$</code> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <code>ExpectedResult=%</code>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> – <code>PASS</code>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail. – <code>FAIL</code>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail. ▶ <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points. ▶ <code>Wait=%, %</code>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10, 40</code> (retry the test every 10 seconds, but time out the test after 40 seconds). |

- Comments** This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.
- Example** This example captures the properties of the month calendar control labeled “Select a Date” and compares them to the recorded baseline in verification point CALENDAR1.
- ```
Result = CalendarVP (CompareProperties, "Label=Select a Date",
"VP=CALENDAR1")
```
- See Also** Calendar

## Call Statement

---

**Description** Transfers control to a sub procedure or function.

**Syntax**

Syntax A **Call** *subprocedure-name* [(*argumentlist*)]

Syntax B *subprocedure-name* *argumentlist*

| Syntax Element           | Description                                               |
|--------------------------|-----------------------------------------------------------|
| <i>subprocedure-name</i> | The name of the sub procedure or function to call.        |
| <i>argumentlist</i>      | The arguments for the sub procedure or function (if any). |

**Comments** Use the `Call` statement to call a sub procedure or function written in SQABasic or to call C procedures in a DLL. These C procedures must be described in a `Declare` statement or be implicit in the application.

If a procedure accepts named arguments, you can use the names to specify the argument and its value. Order is not important. For example, if a procedure is defined as follows:

```
Sub mysub(aa, bb, optional cc, optional dd)
```

The following calls to this procedure are all equivalent:

```
call mysub(1, 2, , 4)
mysub aa := 1, bb := 2, dd :=4
call mysub(aa := 1, dd:=4, bb := 2)
mysub 1, 2, dd:=4
```

## Call

Note that the syntax for named arguments is as follows:

```
argname := argvalue
```

where *argname* is the name for the argument as supplied in the `Sub` or `Function` statement and *argvalue* is the value to assign to the argument when you call it. The advantage to using named arguments is that you do not have to remember the order specified in the procedure's original definition, and if the procedure takes optional arguments, you do not need to include commas (,) for arguments that you leave out.

The procedures that use named arguments include:

- ▶ All functions defined with the `Function` statement.
- ▶ All sub procedures defined with the `Sub` statement.
- ▶ All procedures declared with `Declare` statement.
- ▶ Many built-in functions and statements (such as `InputBox`).
- ▶ Some externally registered DLL functions and methods.

Arguments are passed by reference to procedures written in SQABasic. If you pass a variable to a procedure that modifies its corresponding formal parameter, and you do not want to have your variable modified, enclose the variable in parentheses in the `Call` statement. This will tell SQABasic to pass a copy of the variable. Note that this will be less efficient, and should not be done unless necessary.

When a variable is passed to a procedure that expects its argument by reference, the variable must match the exact type of the formal parameter of the function. (This restriction does not apply to expressions or Variants.)

When calling an external DLL procedure, arguments can be passed by value rather than by reference. This is specified either in the `Declare` statement, the `Call` itself, or both, using the `ByVal` keyword. If `ByVal` is specified in the declaration, then the `ByVal` keyword is optional in the call. If present, it must precede the value. If `ByVal` was not specified in the declaration, it is illegal in the call unless the data type specified in the declaration was `Any`.

## Example

This example calls a sub procedure named `CREATEFILE` to open a file, write the numbers 1 to 10 in it and leave it open. The calling procedure then checks the file's mode. If the mode is 1 (open for Input) or 2 (open for Output), the procedure closes the file.

```
Declare Sub createfile()
Sub main
 Dim filemode as Integer
 Dim attrib as Integer
 Call createfile
 attrib=1
 filemode=FileAttr(1,attrib)
```

```

 If filemode=1 or 2 then
 MsgBox "File was left open. Closing now."
 Close #1
 End If
 Kill "C:\TEMP001"
End Sub

Sub createfile()
 Rem Put the numbers 1-10 into a file
 Dim x as Integer
 Open "C:\TEMP001" for Output as #1
 For x=1 to 10
 Write #1, x
 Next x
End Sub

```

See Also      Declare

## CallScript

Utility Command



**Description**      Causes a script to be executed from within the currently-running script.

**Syntax**            **CallScript** *script\$*

| Syntax Element  | Description                                   |
|-----------------|-----------------------------------------------|
| <i>script\$</i> | Name of the script to be called and executed. |

**Comments**        This event control statement causes a script to call another script. The called, or nested, script executes completely, and then control returns to the calling script. The calling script is suspended while the called script finishes. You can nest scripts up to 16 levels deep (the original script plus up to 15 scripts below it).

This statement corresponds to the Call Script option in the Robot Insert menu. During script recording, use this option to insert a call to another script. You can select the Run Now check box to execute the called script while recording, or deselect it to execute the called script at playback only.

**Example**            This example plays back the script `MyScript` from within the currently executing script.

```
CallScript "MyScript"
```

See Also            None.

CancelButton

## CancelButton

Statement

---

**Description** Sets the position and size of a Cancel button in a dialog box.

**Syntax** `CancelButton x, y, dx, dy [, .id]`

| Syntax Element      | Description                                                                            |
|---------------------|----------------------------------------------------------------------------------------|
| <code>x, y</code>   | The position of the Cancel button relative to the upper left corner of the dialog box. |
| <code>dx, dy</code> | The width and height of the button.                                                    |
| <code>.id</code>    | An optional identifier for the button.                                                 |

**Comments** A `dy` value of 14 can usually accommodate text in the system font.

`.Id` is used by the dialog statements that act on this control. If you use the `Dialog` statement to display the dialog box and the user clicks Cancel, the box is removed from the screen and an Error 102 is triggered. If you use the `Dialog` function to display the dialog box, the function will return 0 and no error occurs.

Use the `CancelButton` statement only between a `Begin Dialog` and an `End Dialog` statement.

**Example** This example defines a dialog box with a combo box and buttons.

```
Sub main
 Dim fchoices as String
 fchoices="File1" & Chr(9) & "File2" & Chr(9) & "File3"
 Begin Dialog UserDialog 185, 94, "SQABasic Dialog Box"
 Text 9, 5, 69, 10, "Filename:", .Text1
 DropComboBox 9, 17, 88, 71, fchoices, .ComboBox1
 ButtonGroup .ButtonGroup1
 OKButton 113, 14, 54, 13
 CancelButton 113, 33, 54, 13
 PushButton 113, 57, 54, 13, "Help", .Push1
 End Dialog
 Dim mydialog as UserDialog
 On Error Resume Next
 Dialog mydialog
 If Err=102 then
 MsgBox "Dialog box canceled."
 End If
End Sub
```

**See Also**

|                           |                           |                             |
|---------------------------|---------------------------|-----------------------------|
| <code>Begin Dialog</code> | <code>ComboBox</code>     | <code>OptionButton</code>   |
| <code>End Dialog</code>   | <code>DropComboBox</code> | <code>OptionGroup</code>    |
| <code>Button</code>       | <code>DropListBox</code>  | <code>Picture</code>        |
| <code>CancelButton</code> | <code>GroupBox</code>     | <code>StaticComboBox</code> |
| <code>Caption</code>      | <code>ListBox</code>      | <code>Text</code>           |
| <code>CheckBox</code>     | <code>OKButton</code>     | <code>TextBox</code>        |

## Caption Statement

---

**Description** Defines the title of a dialog box.

**Syntax** `Caption text$`

| Syntax Element      | Description                                                 |
|---------------------|-------------------------------------------------------------|
| <code>text\$</code> | A string expression containing the title of the dialog box. |

**Comments** Use the `Caption` statement only between a `Begin Dialog` and an `End Dialog` statement.  
If no `Caption` statement is specified for the dialog box, a default caption is used.

**Example** This example defines a dialog box with a combination list box and three buttons. The `Caption` statement changes the dialog box title to `Example-Caption Statement`.

```
Sub main
 Dim fchoices as String
 fchoices="File1" & Chr(9) & "File2" & Chr(9) & "File3"
 Begin Dialog UserDialog 185, 94
 Caption "Example-Caption Statement"
 Text 9, 5, 69, 10, "Filename:", .Text1
 DropComboBox 9, 17, 88, 71, fchoices, .ComboBox1
 ButtonGroup .ButtonGroup1
 OKButton 113, 14, 54, 13
 CancelButton 113, 33, 54, 13
 PushButton 113, 57, 54, 13, "Help", .Push1
 End Dialog
 Dim mydialog as UserDialog
 On Error Resume Next
 Dialog mydialog
 If Err=102 then
 MsgBox "Dialog box canceled."
 End If
End Sub
```

**See Also**

|                           |                           |                             |
|---------------------------|---------------------------|-----------------------------|
| <code>Begin Dialog</code> | <code>ComboBox</code>     | <code>OptionButton</code>   |
| <code>End Dialog</code>   | <code>DropComboBox</code> | <code>OptionGroup</code>    |
| <code>Button</code>       | <code>DropListBox</code>  | <code>Picture</code>        |
| <code>CancelButton</code> | <code>GroupBox</code>     | <code>StaticComboBox</code> |
| <code>Caption</code>      | <code>ListBox</code>      | <code>Text</code>           |
| <code>CheckBox</code>     | <code>OKButton</code>     | <code>TextBox</code>        |

CCur

## CCur Function

---

**Description**    Converts an expression to the data type Currency.

**Syntax**            **CCur**(*expression*)

| Syntax Element    | Description                                |
|-------------------|--------------------------------------------|
| <i>expression</i> | Any expression that evaluates to a number. |

**Comments**        **CCur** accepts any type of *expression*. Numbers that do not fit in the Currency data type result in an Overflow error. Strings that cannot be converted result in a Type Mismatch error. Variants containing null result in an Illegal Use of Null error.

**Example**            This example converts a yearly payment on a loan to a currency value with four decimal places. A subsequent Format statement formats the value to two decimal places before displaying it in a message box.

```
Sub main
Dim aprate, totalpay,loanpv
Dim loanfv, due, monthlpay
Dim yearlpay, msgtext
loanpv=InputBox("Enter the loan amount: ")
aprate=InputBox("Enter the annual percentage rate: ")
If aprate > 1 then
 Aprate = aprate/100
End If
aprate=aprate/12
totalpay=InputBox("Enter the total number of pay periods: ")
loanfv=0
Rem Assume payments are made at end of month
due=0
monthlpay=Pmt(aprate,totalpay,-loanpv,loanfv,due)
yearlpay=CCur(monthlpay*12)
msgtext="The yearly payment is: " & Format(yearlpay,"Currency")
MsgBox msgtext
End Sub
```

**See Also**            CDb1            CStr  
                  CInt            CVar  
                  CLng          CVDate  
                  CSng

## CDBl

### Function

---

**Description**    Converts an expression to the data type Double.

**Syntax**        **CDBl** (*expression*)

| Syntax Element    | Description                                |
|-------------------|--------------------------------------------|
| <i>expression</i> | Any expression that evaluates to a number. |

**Comments**     CDBl accepts any type of *expression*. Strings that cannot be converted to a double-precision floating point result in a `Type Mismatch` error. Variants containing null result in an `Illegal Use of Null` error.

**Example**        This example calculates the square root of 2 as a double-precision floating point value and displays it in scientific notation.

```
Sub main
 Dim value
 Dim msgtext
 value=CDBl(Sqr(2))
 msgtext= "The square root of 2 is: " & Value
 MsgBox msgtext
End Sub
```

**See Also**        CCur        CStr  
 CInt        CVar  
 CLng        CDate  
 CSng

## ChDir

### Statement

---

**Description**    Changes the default directory for the specified drive. .

**Syntax**        **ChDir** *path\$*

| Syntax Element | Description                                                |
|----------------|------------------------------------------------------------|
| <i>path\$</i>  | A string expression identifying the new default directory. |

ChDrive

Comments    The syntax for *path\$* is:

```
[drive:][\]directory[\directory]
```

If the drive argument is omitted, `ChDir` changes the default directory on the current drive. The `ChDir` statement does not change the default drive. To change the default drive, use `ChDrive`.

Example      This example changes the current directory to C:\WINDOWS, if it is not already the default.

```
Sub main
 Dim newdir as String
 newdir="c:\windows"
 If CurDir <> newdir then
 ChDir newdir
 End If
 MsgBox "The default directory is now: " & newdir
End Sub
```

See Also      `ChDrive`          `MkDir`  
                 `CurDir`          `Rmdir`  
                 `Dir`

## ChDrive Statement

---

Description    Changes the default drive.

**ChDrive** *drive\$*

| Syntax Element | Description                                            |
|----------------|--------------------------------------------------------|
| <i>drive\$</i> | A string expression designating the new default drive. |

Comments      This drive must exist and must be within the range specified by the LASTDRIVE statement in the CONFIG.SYS file. If a null argument ( " ") is supplied, the default drive remains the same. If the *drive\$* argument is a string, `ChDrive` uses the first letter only. If the argument is omitted, an error message is produced. To change the current directory on a drive, use `ChDir`.



**Example** This example changes the default drive to A:.

```
Sub main
 Dim newdrive as String
 newdrive="A:"
 If Left(CurDir,2) <> newdrive then
 ChDrive newdrive
 End If
 MsgBox "The default drive is now " & newdrive
End Sub
```

**See Also** ChDir            Mkdir  
 CurDir            Rmdir  
 Dir

## CheckBox Statement

---

**Description** Creates a check box in a dialog box.

**Syntax** **CheckBox** *x, y , dx, dy, text\$, .field*

| Syntax Element | Description                                                                                                         |
|----------------|---------------------------------------------------------------------------------------------------------------------|
| <i>x, y</i>    | The upper left corner coordinates of the check box, relative to the upper left corner of the dialog box.            |
| <i>dx</i>      | The sum of the widths of the check box and <i>text\$</i> .                                                          |
| <i>dy</i>      | The height of <i>text\$</i> .                                                                                       |
| <i>text\$</i>  | The title shown to the right of the check box.                                                                      |
| <i>.field</i>  | The name of the dialog-record field that will hold the current check box setting (0=unchecked, -1=grey, 1=checked). |

**Comments** The *x* argument is measured in 1/4 system-font character-width units. The *y* argument is measured in 1/8 system-font character-height units. (See `Begin Dialog` for more information.)

Because proportional spacing is used, the *dx* argument width will vary with the characters used. To approximate the width, multiply the number of characters in the *text\$* field (including blanks and punctuation) by 4 and add 12 for the check box.

A *dy* value of 12 is standard, and should cover typical default fonts. If larger fonts are used, the value should be increased. As the *dy* number grows, the check box and the accompanying text will move down within the dialog box.

## CheckBox (User Action Command)

If the width of the *text\$* field is greater than *dx*, trailing characters will be truncated. If you want to include underlined characters so that the check box selection can be made from the keyboard, precede the character to be underlined with an ampersand (&).

SQABasic treats any other value of *.field* the same as a 1. The *.field* argument is also used by the dialog statements that act on this control.

Use the **CheckBox** statement only between a **Begin Dialog** and an **End Dialog** statement.

**Example** This example defines a dialog box with a combination list box, a check box, and three buttons.

```
Sub main
 Dim ComboBox1() as String
 ReDim ComboBox1(0)
 ComboBox1(0)=Dir("C:*. *")
 Begin Dialog UserDialog 166, 76, "SQABasic Dialog Box"
 Text 9, 3, 69, 13, "Filename:", .Text1
 DropComboBox 9, 14, 81, 119, ComboBox1(), .ComboBox1
 CheckBox 10, 39, 62, 14, "List .TXT files", .CheckBox1
 OKButton 101, 6, 54, 14
 CancelButton 101, 26, 54, 14
 PushButton 101, 52, 54, 14, "Help", .Push1
 End Dialog
 Dim mydialog as UserDialog
 On Error Resume Next
 Dialog mydialog
 If Err=102 then
 MsgBox "Dialog box canceled."
 End If
End Sub
```

|          |              |              |                |
|----------|--------------|--------------|----------------|
| See Also | Begin Dialog | ComboBox     | OptionButton   |
|          | End Dialog   | DropComboBox | OptionGroup    |
|          | Button       | DropListBox  | Picture        |
|          | CancelButton | GroupBox     | StaticComboBox |
|          | Caption      | Listbox      | Text           |
|          | CheckBox     | OKButton     | TextBox        |

## CheckBox

User Action Command



**Description** Performs an action on a check box control.

Syntax

**CheckBox** *action%*, *recMethod\$*

| Syntax Element     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i>     | <p>The following mouse action:</p> <p><i>MouseClick</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). Coordinates are not required. See Appendix E for a list of mouse click values.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <i>recMethod\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <i>HTMLId=\$</i>. The text from the ID attribute of the HTML object.</li> <li>▶ <i>HTMLText=\$</i>. The text of a check box in a Web page INPUT form element. The text is from the Value attribute of the INPUT tag.</li> <li>▶ <i>HTMLTitle=\$</i>. The text from the Title attribute of the HTML object.</li> <li>▶ <i>ID=%</i>. The object's internal Windows ID.</li> <li>▶ <i>Index=%</i>. The number of the object among all objects identified with the same base recognition method. Typically, <i>Index</i> is used after another recognition method qualifier — for example, <i>Name=\$; Index=%</i>.</li> <li>▶ <i>JavaText=\$</i>. A label that identifies the object in the user interface.</li> <li>▶ <i>Name=\$</i>. A unique name that a developer assigns to an object to identify the object in the development environment. For example, the object name for a command button might be <i>Command1</i>.</li> <li>▶ <i>ObjectIndex=%</i>. The number of the object among all objects of the same type in the same window.</li> <li>▶ <i>State=\$</i>. An optional qualifier for any other recognition method. There are two possible values for this setting: <i>Enabled</i> and <i>Disabled</i>. The default state is the state of the current context window (as set in the most recent <i>Window SetContext</i> statement), or <i>Enabled</i> if the state has not been otherwise declared.</li> <li>▶ <i>Text=\$</i>. The text displayed on the object.</li> <li>▶ <i>Type=\$</i>. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The <i>Type</i> qualifier uses the following form: <i>Type=\$; recMethod=\$</i>. Parent/child values are separated by a backslash and semicolons (<i>;\;</i>).</li> <li>▶ <i>VisualText=\$</i>. An optional setting used to identify an object by its visible text. It is for user clarification only and does not affect object recognition.</li> </ul> |

CheckBoxVP

Comments None.

Example This example clicks the check box with the Visual Basic object name of "Overdraft".

```
CheckBox Click, "Name=Overdraft"
```

This example clicks the check box with a Value attribute of 2. The check box is located within the Web page frame named Main.

```
CheckBox Click,
"Type=HTMLFrame;HTMLId=Main;\;Type=CheckBox;HTMLText=2"
```

See Also Label Radiobutton  
PushButton

## CheckBoxVP

Verification Point Command



Description Establishes a verification point for a check box control.

Syntax *Result* = **CheckBoxVP** (*action%*, *recMethod\$*, *parameters\$*)

| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i> | <p>The type of verification to perform. Valid values:</p> <ul style="list-style-type: none"><li>▶ <b>CompareData</b>. Captures the contents or HTML text of the object and compares it to a recorded baseline. <i>parameters\$ VP</i> is required; <b>ExpectedResult</b> and <b>Wait</b> are optional.</li><li>▶ <b>CompareNumeric</b>. Captures the numeric value of the text of the object and compares it to the value of <i>parameters\$ Value</i> or <i>Range</i>. <i>parameters\$ VP</i> and either <b>Value</b> or <b>Range</b> are required; <b>ExpectedResult</b> and <b>Wait</b> are optional.</li><li>▶ <b>CompareProperties</b>. Captures object properties information for the object and compares it to a recorded baseline. <i>parameters\$ VP</i> is required; <b>ExpectedResult</b> and <b>Wait</b> are optional.</li><li>▶ <b>CompareText</b>. Captures the text of the object and compares it to a recorded baseline. <i>parameters\$ VP</i> and <b>Type</b> are required; <b>ExpectedResult</b> and <b>Wait</b> are optional.</li><li>▶ <b>VerifyIsBlank</b>. Checks that the object has no text. <i>parameters\$ VP</i> is required; <b>ExpectedResult</b> and <b>Wait</b> are optional.</li></ul> |





| Syntax Element       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>recMethod</i> \$  | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <b>HTMLId=\$</b>. The text from the ID attribute of the HTML object.</li> <li>▶ <b>HTMLText=\$</b>. The visible text of a check box in a Web page INPUT form element. The text is from the Name attribute of the INPUT tag.</li> <li>▶ <b>HTMLTitle=\$</b>. The text from the Title attribute of the HTML object.</li> <li>▶ <b>ID=%</b>. The object's internal Windows ID.</li> <li>▶ <b>Index=%</b>. The number of the object among all objects identified with the same base recognition method. Typically, <b>Index</b> is used after another recognition method qualifier — for example, <b>Name=\$; Index=%</b>.</li> <li>▶ <b>JavaText=\$</b>. A label that identifies the object in the user interface.</li> <li>▶ <b>Name=\$</b>. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be <b>Command1</b>.</li> <li>▶ <b>ObjectIndex=%</b>. The number of the object among all objects of the same type in the same window.</li> <li>▶ <b>Text=\$</b>. The text displayed on the object.</li> <li>▶ <b>Type=\$</b>. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The <b>Type</b> qualifier uses the following form: <b>Type=\$; recMethod=\$</b>. Parent/child values are separated by a backslash and semicolons (<b>i \ i</b>).</li> </ul> |
| <i>parameters</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <b>ExpectedResult=%</b>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> <li>– <b>PASS</b>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail.</li> <li>– <b>FAIL</b>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail.</li> </ul> </li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |



## CheckBoxVP

▶ ▶ ▶

| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                | <ul style="list-style-type: none"> <li>▶ <code>Range=&amp; , &amp;</code>. Used with the action <code>CompareNumeric</code> when a numeric range comparison is being performed, as in <code>Range=2 , 12</code> (test for numbers in this range). The values are inclusive.</li> <li>▶ <code>Type=\$</code>. Specifies the verification method to use for <code>CompareText</code> actions. The possible values are: <code>CaseSensitive</code>, <code>CaseInsensitive</code>, <code>FindSubStr</code>, <code>FindSubStrI</code> (case insensitive), and <code>UserDefined</code>. See <i>Comments</i> for more information. If <code>UserDefined</code> is specified, two additional parameters are required: <ul style="list-style-type: none"> <li>- <code>DLL=\$</code>. The full path and file name of the library that contains the function</li> <li>- <code>Function=\$</code>. The name of the custom function to use in comparing the text</li> </ul> </li> <li>▶ <code>Value=&amp;</code>. Used with the action <code>CompareNumeric</code> when performing a numeric equivalence comparison, as in <code>Value=25</code> (test against the value 25).</li> <li>▶ <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points.</li> <li>▶ <code>Wait=% , %</code>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10 , 40</code> (retry the test every 10 seconds, but time out the test after 40 seconds).</li> </ul> |

**Comments** This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

With the `Type=$` parameter, `CaseSensitive` and `CaseInsensitive` require a full match between the current baseline text and the text captured during playback. With `FindSubStr` and `FindSubStrI`, the current baseline can be a substring of the text captured during playback. The substring can appear anywhere in the playback text. To modify the current baseline text, double-click the verification point name in the Robot Asset pane (to the left of the script).

**Example** This example captures the properties of the check box identified by the text `Read Only` and compares them to the recorded baseline in verification point `VP TEN`. At playback, the comparison is retried every 6 seconds and times out after 30 seconds.

```
Result = CheckBoxVP (CompareProperties, "Text=Read Only",
 "VP=VP TEN;Wait=6,30")
```

This example captures the properties of the check box with a Name attribute of Check1. The check box is located within the Web page frame named Main. CheckBoxVP compares the properties to the recorded baseline in verification point CHKVP TEN. At playback, the comparison is retried every 2 seconds and times out after 30 seconds.

```
Result = CheckBoxVP (CompareData,
 "Type=HTMLFrame;HTMLId=Main;\;Type=CheckBox;Name=Check1",
 "VP=CHKVP TEN;Wait=2,30")
```

See Also

LabelVP  
 PushButton  
 RadioButtonVP

## Chr Function

---

**Description** Returns a one-character string corresponding to a character code.

**Syntax** **Chr**[\$]( *charcode%* )

| Syntax Element   | Description                                                                                                              |
|------------------|--------------------------------------------------------------------------------------------------------------------------|
| \$               | Optional. If specified, the return type is String. If omitted, the function will return a Variant of VarType 8 (String). |
| <i>charcode%</i> | A number representing the character to be returned.                                                                      |

**Comments** To obtain a byte representing a given character, use ChrB.

**Example** This example displays the character equivalent for an ASCII code between 65 and 122 typed by the user.

```
Sub main
 Dim numb as Integer
 Dim msgtext
 Dim out
 out=0
 Do Until out
 numb=InputBox("Type a number between 65 and 122:")
 If Chr$(numb)>="A" AND Chr$(numb)<="Z" OR
 Chr$(numb)>="a" AND Chr$(numb)<="z" then
 msgtext="The letter for the number " & numb &
 " is: " & Chr$(numb)
 out=1
 ElseIf numb=0 then
 Exit Sub
 End If
 Loop
```

## CInt

```
 Else
 Beep
 msgtext="Does not convert to a character; try again."
 End If
 MsgBox msgtext
 Loop
End Sub
```

See Also

|      |      |        |
|------|------|--------|
| Asc  | CLng | CVDate |
| CCur | CSng | Format |
| Cdbl | CStr | Val    |
| CInt | CVar |        |

## CInt

### Function

---

**Description** Converts an expression to the data type Integer by rounding.

**Syntax** **CInt**(*expression*)

| Syntax Element    | Description                                   |
|-------------------|-----------------------------------------------|
| <i>expression</i> | Any expression that can evaluate to a number. |

**Comments** After rounding, the resulting number must be within the range of -32767 to 32767, or an error occurs.

Strings that cannot be converted to an integer result in a `Type Mismatch` error.

Variants containing null result in an `Illegal Use of Null` error.

**Example** This example calculates the average of ten golf scores.

```
Sub main
 Dim score As Integer
 Dim x, sum
 Dim msgtext
 Let sum=0
 For x=1 to 10
 score=InputBox("Enter golf score #"&x &":")
 sum=sum+score
 Next x
 msgtext = "Your average is: " & Format(CInt(sum/(x-1)))
 MsgBox msgtext
End Sub
```

See Also

|      |        |
|------|--------|
| CCur | CStr   |
| Cdbl | CVar   |
| CLng | CVDate |
| CSng |        |



## Class List

---

|             |                                                                                                                                                     |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| Description | The <code>Object</code> class can be used in a <code>Dim</code> statement, a <code>TypeOf</code> expression, or with the <code>New</code> operator. |
| Syntax      | <b>Object</b>                                                                                                                                       |
| Comments    | Provides access to OLE2 automation.                                                                                                                 |
| Example     | None.                                                                                                                                               |
| See Also    | None.                                                                                                                                               |

## Clipboard

---

**Description** The Windows Clipboard can be accessed directly in your program to enable you to get text from and put text into other applications that support the Clipboard.

**Syntax**

```
Clipboard.Clear
Clipboard.GetText()
Clipboard.SetText string$
Clipboard.GetFormat(1)
```

| Syntax Element  | Description                                                                                                             |
|-----------------|-------------------------------------------------------------------------------------------------------------------------|
| <i>string\$</i> | A string or string expression containing the text to send to the Clipboard. Used with the <code>.SetText</code> method. |

**Comments** The following Clipboard methods are supported:

| Method                  | Description                                                                                                                                     |
|-------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>.Clear</code>     | Clears the contents of the Clipboard.                                                                                                           |
| <code>.GetText</code>   | Returns a text string from the Clipboard.                                                                                                       |
| <code>.SetText</code>   | Puts a text string to the Clipboard.                                                                                                            |
| <code>.GetFormat</code> | This method always takes the argument 1. Returns TRUE (non-0) if the format of the item on the Clipboard is text. Otherwise, returns FALSE (0). |

Data on the Clipboard is lost when another set of data of the same format is placed on the Clipboard (either through code or a menu command).

## ClipboardVP

**Example** This example places the text string `Hello, world` on the Clipboard.

```
Sub main
 Dim mytext as String
 mytext="Hello, World"
 Clipboard.SetText mytext
 MsgBox "The text: '" & mytext & "' added to the Clipboard."
End Sub
```

**See Also** None.

## ClipboardVP

Verification Point Command

»»SQAS»

**Description** Establishes a verification point for the contents of the Windows Clipboard.

```
Result = ClipboardVP (action%, "", parameters$)
```

| Syntax Element      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i>      | The type of verification to perform. Valid value: <ul style="list-style-type: none"><li>► <code>Compare</code>. Captures the text of the Clipboard and compares it to a recorded baseline. <i>parameters\$ VP</i> is required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li></ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| " "                 | The second argument is always left blank.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <i>parameters\$</i> | Valid values: <ul style="list-style-type: none"><li>► <code>ExpectedResult=%</code>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values:<ul style="list-style-type: none"><li>– <code>PASS</code>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail.</li><li>– <code>FAIL</code>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail.</li></ul></li><li>► <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points.</li><li>► <code>Wait=%, %</code>. A Wait State that specifies the verification point's Retry value and Timeout value, as in <code>Wait=1, 30</code> where 1 indicates the verification point is to be retried every second but timed-out after 30 seconds.</li></ul> |

- Comments** This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.  
Only textual data on the Clipboard can be compared.
- Example** This example captures the contents of the Clipboard and compares it to a recorded baseline in verification point CBOARDA.
- ```
Result = ClipboardVP (Compare, "", "VP=CBOARDA")
```
- See Also** None.

CLng Function

Description Converts an expression to the data type Long by rounding.

Syntax **CLng**(*expression*)

| Syntax Element | Description |
|-------------------|---|
| <i>expression</i> | Any expression that can evaluate to a number. |

Comments After rounding, the resulting number must be within the range of -2,147,483,648 to 2,147,483,647, or an error occurs.

Strings that cannot be converted to a long result in a `Type Mismatch` error.
Variants containing null result in an `Illegal Use of Null` error.

Example This example divides the US national debt by the number of people in the country to find the amount of money each person would have to pay to wipe it out. This figure is converted to a Long integer and formatted as Currency.

```
Sub Main
  Dim debt As Single
  Dim msgtext
  Const Populace = 250000000
  debt=InputBox("Enter the current US national debt:")
  msgtext = "The debt per citizen is: "
  msgtext = msgtext + Format(CLng(Debt/Populace), "Currency")
  MsgBox msgtext
End Sub
```

See Also CCur CStr
CDBl CVar
CInt CDate
CSng

Close

Close Statement

Description Closes a file, concluding input/output to that file.

Syntax **Close** [[#]*filenumber%*[, [#]*filenumber%*]]

| Syntax Element | Description |
|--------------------|---|
| <i>filenumber%</i> | An integer expression identifying the number of the file to close. If omitted, all open files are closed. |

Comments *filenumber%* is the number assigned to the file in the `Open` statement. Once a `Close` statement is executed, the association of a file with *filenumber%* is ended, and the file can be reopened with the same or a different file number.

When the `Close` statement is used, the final output buffer is written to the operating system buffer for that file. `Close` frees all buffer space associated with the closed file. Use the `Reset` statement so that the operating system will flush its buffers to disk.

This example opens a file for Random access, gets the contents of one variable, and closes the file again. The sub procedure `CREATEFILE` creates the file `C:\TEMP001` used by the main sub procedure.

```
Declare Sub createfile()
Sub main
  Dim acctno as String*3
  Dim recno as Long
  Dim msgtext as String
  Call createfile
  recno=1
  newline=Chr(10)
  Open "C:\TEMP001" For Random As #1 Len=3
  msgtext="The account numbers are:" & newline & newline
  Do Until recno=11
    Get #1,recno,acctno
    msgtext=msgtext & acctno
    recno=recno+1
  Loop
  MsgBox msgtext
  Close #1
  Kill "C:\TEMP001"
End Sub

Sub createfile()
  Rem Put the numbers 1-10 into a file
  Dim x as Integer
  Open "C:\TEMP001" for Output as #1
  For x=1 to 10
    Write #1, x
  Next x
  Close #1
End Sub
```

See Also `Open`
 `Reset`
 `Stop`

ComboBox Statement

Description Creates a combination text box and list box in a dialog box.

Syntax Syntax A **ComboBox** *x, y, dx, dy, text\$, .field*

 Syntax B **ComboBox** *x, y, dx, dy, stringarray\$, .field*

| Syntax Element | Description |
|----------------------|---|
| <i>x, y</i> | The upper left corner coordinates of the list box, relative to the upper left corner of the dialog box. |
| <i>dx, dy</i> | The width and height of the combo box in which the user enters or selects text. |
| <i>text\$</i> | A string containing the selections for the combo box. |
| <i>stringarray\$</i> | An array of dynamic strings for the selections in the combo box. |
| <i>.field</i> | The name of the dialog-record field that will hold the text string entered in the text box or chosen from the list box. |

Comments The *x* argument is measured in 1/4 system-font character-width units. The *y* argument is measured in 1/8 system-font character-width units. (See `Begin Dialog` for more information.)

The *text\$* argument must be defined, using a `Dim` statement, before the `Begin Dialog` statement is executed. The arguments in the *text\$* string are entered as shown in the following example:

```
dimname="listchoice"+Chr$(9)+"listchoice"+Chr$(9)+"listchoice"...
```

The string in the text box will be recorded in the field designated by the *.field* argument when the OK button (or any push button other than Cancel) is pushed. The *field* argument is also used by the dialog statements that act on this control.

Use the `ComboBox` statement only between a `Begin Dialog` and an `End Dialog` statement.

ComboBox (User Action Command)

Example This example defines a dialog box containing a combo box and three buttons.

```
Sub main
  Dim ComboBox1() as String
  ReDim ComboBox1(0)
  ComboBox1(0)=Dir("C:\*.*.")
  Begin Dialog UserDialog 166, 142, "SQABasic Dialog Box"
    Text 9, 3, 69, 13, "Filename:", .Text1
    ComboBox 9, 14, 81, 119, ComboBox1(), .ComboBox1
    OKButton 101, 6, 54, 14
    CancelButton 101, 26, 54, 14
    PushButton 101, 52, 54, 14, "Help", .Push1
  End Dialog
  Dim mydialog as UserDialog
  On Error Resume Next
  Dialog mydialog
  If Err=102 then
    MsgBox "Dialog box canceled."
  End If
End Sub
```

See Also

| | | |
|--------------|--------------|----------------|
| Begin Dialog | ComboBox | OptionGroup |
| End Dialog | DropComboBox | Picture |
| Button | DropListBox | StaticComboBox |
| ButtonGroup | GroupBox | Text |
| CancelButton | ListBox | TextBox |
| Caption | OKButton | |
| CheckBox | OptionButton | |

ComboBox

User Action Command



Description Performs an action on a combo box control.

Syntax **ComboBox** *action%*, *recMethod\$*, *parameters\$*

| Syntax Element | Description |
|----------------|---|
| <i>action%</i> | One of these mouse actions: <ul style="list-style-type: none">► MakeSelection. Selects the specified item from a Java combo box. Used only for the Java environment. <i>recMethod\$</i> must contain one of the Java recognition methods, and <i>parameters\$</i> must contain either Text or Index.► MouseClicked. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> must contain Coords=x,y. |



▶ ▶ ▶

| Syntax Element | Description |
|--------------------|---|
| | <p>▶ <i>MouseDown</i>. The dragging of the mouse while mouse buttons and/or shifting keys (Ctrl, Alt, Shift) are pressed. When <i>action%</i> contains a mouse-drag value, <i>parameters\$</i> must contain <i>Coords=x1,y1,x2,y2</i>. See Appendix E for a list of mouse click and drag values.</p> |
| <i>recMethod\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <i>HTMLId=\$</i>. The text from the ID attribute of the HTML object. ▶ <i>HTMLText=\$</i>. The visible text of a Web page SELECT form element. The text is from the Value attribute of the OPTION tag. ▶ <i>HTMLTitle=\$</i>. The text from the Title attribute of the HTML object. ▶ <i>ID=%</i>. The object's internal Windows ID. ▶ <i>Index=%</i>. The number of the object among all objects identified with the same base recognition method. Typically, <i>Index</i> is used after another recognition method qualifier — for example, <i>Name=\$; Index=%</i>. ▶ <i>JavaText=\$</i>. A label that identifies the object in the user interface. ▶ <i>Label=\$</i>. The text of the label object that immediately precedes the combo box in the Z order of windows. ▶ <i>Name=\$</i>. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be <i>Command1</i>. ▶ <i>ObjectIndex=%</i>. The number of the object among all objects of the same type in the same window. ▶ <i>State=\$</i>. An optional qualifier for any other recognition method. There are two possible values for this setting: <i>Enabled</i> and <i>Disabled</i>. The default state is the state of the current context window (as set in the most recent <i>Window SetContext</i> command), or <i>Enabled</i> if the state has not been otherwise declared. ▶ <i>Type=\$</i>. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The <i>Type</i> qualifier uses the following form: <i>Type=\$; recMethod=\$</i>. Parent/child values are separated by a backslash and semicolons (<i>;\;</i>). ▶ <i>VisualText=\$</i>. An optional setting used to identify an object by its prior label. It is for user clarification only and does not affect object recognition. |

▶ ▶ ▶

ComboBoxVP

▶ ▶ ▶

| Syntax Element | Description |
|----------------------|---|
| <i>parameters</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none">▶ <i>Coords=x,y</i>. If <i>action%</i> is a mouse click, specifies the coordinates of the click, relative to the top left of the object.▶ <i>Coords=x1,y1,x2,y2</i>. If <i>action%</i> is a mouse drag, specifies the coordinates, where <i>x1,y1</i> are the starting coordinates of the drag, and <i>x2,y2</i> are the ending coordinates. The coordinates are relative to the top left of the object.▶ <i>Index=%</i>. If <i>action%</i> is <i>MakeSelection</i>, identifies the index of an item in the list.▶ <i>Text=\$</i>. If <i>action%</i> is <i>MakeSelection</i>, identifies the text of an item in the list. |

Comments None.

Example This example clicks the second combo box in the window (*ObjectIndex=2*) at *x,y* coordinates of 49,14. The combo box is preceded by a label with the text "Name:".

```
ComboBox Click, "ObjectIndex=2;VisualText=Name:", "Coords=49,14"
```

This example clicks the combo box with a *Name* attribute of *Selectlist*. The combo box is located within the Web page frame named *Main*.

```
ComboBox Click, "Type=HTMLFrame;HTMLId=Main;\;Type=ComboBox;  
Name=Selectlist", ""
```

See Also

| | |
|---------------------------|----------------------|
| <code>ComboEditBox</code> | <code>EditBox</code> |
| <code>ComboListBox</code> | <code>ListBox</code> |

ComboBoxVP

Verification Point Command

▶▶SQA▶

Description Establishes a verification point for a combo box control.

Syntax *Result* = **ComboBoxVP** (*action%*, *recMethod*\$, *parameters*\$)

| Syntax Element | Description |
|--------------------|---|
| <i>action%</i> | <p>The type of verification to perform. Valid values:</p> <ul style="list-style-type: none"> ▶ Compare. Captures the entire textual contents of the object into a grid and compares it to a recorded baseline. <i>parameters\$ VP</i> is required; <i>ExpectedResult</i> and <i>Wait</i> are optional. ▶ CompareData. Captures the contents or HTML text of the object and compares it to a recorded baseline. <i>parameters\$ VP</i> is required; <i>ExpectedResult</i> and <i>Wait</i> are optional. ▶ CompareNumeric. Captures the numeric value of the text of the object and compares it to the value of <i>parameters\$ Value</i> or <i>Range</i>. <i>parameters\$ VP</i> and either <i>Value</i> or <i>Range</i> are required; <i>ExpectedResult</i> and <i>Wait</i> are optional. ▶ CompareProperties. Captures object properties information for the object and compares it to a recorded baseline. <i>parameters\$ VP</i> is required; <i>ExpectedResult</i> and <i>Wait</i> are optional. ▶ CompareText. Captures the text of the object and compares it to a recorded baseline. <i>parameters\$ VP</i> and <i>Type</i> are required; <i>ExpectedResult</i> and <i>Wait</i> are optional. ▶ VerifyIsBlank. Checks that the object has no text. <i>parameters\$ VP</i> is required; <i>ExpectedResult</i> and <i>Wait</i> are optional. |
| <i>recMethod\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ HTMLId=\$. The text from the ID attribute of the HTML object. ▶ HTMLText=\$. The visible text of a Web page SELECT form element. The text is from the Value attribute of the OPTION tag. ▶ HTMLTitle=\$. The text from the Title attribute of the HTML object. ▶ ID=%. The object's internal Windows ID. ▶ Index=%. The number of the object among all objects identified with the same base recognition method. Typically, <i>Index</i> is used after another recognition method qualifier — for example, <i>Name=\$; Index=%</i>. ▶ JavaText=\$. A label that identifies the object in the user interface. ▶ Label=\$. The text of the label object that immediately precedes the combo box in the Z order of windows. |

▶ ▶ ▶

▶ ▶ ▶

| Syntax Element | Description |
|---------------------|---|
| | <ul style="list-style-type: none"> ▶ Name=\$. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be Command1. ▶ ObjectIndex=%. The number of the object among all objects of the same type in the same window. ▶ Type=\$. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The Type qualifier uses the following form: Type=\$;recMethod=\$. Parent/child values are separated by a backslash and semicolons (; \ ;). |
| <i>parameters\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ ExpectedResult=%. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> - PASS. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail. - FAIL. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail. ▶ Range=&, &. Used with the action CompareNumeric when a numeric range comparison is being performed, as in Range=2, 12 (test for numbers in this range). The values are inclusive. ▶ Type=\$. Specifies the verification method to use for CompareText actions. The possible values are: CaseSensitive, CaseInsensitive, FindSubStr, FindSubStrI (case insensitive), and UserDefined. See <i>Comments</i> for more information. If UserDefined is specified, two additional parameters are required: <ul style="list-style-type: none"> - DLL=\$. The full path and file name of the library that contains the function - Function=\$. The name of the custom function to use in comparing the text |

▶ ▶ ▶



| Syntax Element | Description |
|----------------|---|
| | <ul style="list-style-type: none"> ▶ Value=&. Used with the action <code>CompareNumeric</code> when a numeric equivalence comparison is being performed, as in <code>Value=25</code> (test against the value 25). ▶ VP=\$. The verification point ID. IDs must be unique within a script. Required for all verification points. ▶ Wait=%,%. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10,40</code> (retry the test every 10 seconds, but time out the test after 40 seconds). |

Comments This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

With the `Type=$` parameter, `CaseSensitive` and `CaseInsensitive` require a full match between the current baseline text and the text captured during playback. With `FindSubStr` and `FindSubStrI`, the current baseline can be a substring of the text captured during playback. The substring can appear anywhere in the playback text. To modify the current baseline text, double-click the verification point name in the Robot Asset pane (to the left of the script).

Example This example captures the properties of the first combo box control in the window (`ObjectIndex=1`) and compares them to the recorded baseline in verification point `VPFIVE`. At playback, the comparison is retried every 4 seconds and times out after 30 seconds.

```
Result = ComboBoxVP (CompareProperties, "ObjectIndex=1",
"VP=VPFIVE; Wait=4,30")
```

This example captures the properties of the select list with a `Name` attribute of `Sselectlist`. The list is located within the Web page frame named `Main`. `ComboBoxVP` compares the properties to the recorded baseline in verification point `SELECTVP1`. At playback, the comparison is retried every 2 seconds and times out after 30 seconds.

```
Result = ComboBoxVP (CompareData,
"Type=HTMLFrame;HTMLId=Main;\;Type=ComboBox;Name=Selectlist",
"VP=SELECTVP1;Wait=2,30")
```

See Also

| | |
|-----------------------------|------------------------|
| <code>ComboEditBoxVP</code> | <code>EditBoxVP</code> |
| <code>ComboListBoxVP</code> | <code>ListBoxVP</code> |

ComboEditBox

User Action Command

Description Performs an action on a combo edit box control.

Syntax **ComboEditBox** *action%*, *recMethod\$*, *parameters\$*

| Syntax Element | Description |
|--------------------|--|
| <i>action%</i> | <p>One of these mouse actions:</p> <ul style="list-style-type: none"> ▶ <i>MouseClicked</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> must contain <i>Coords=x,y</i>. ▶ <i>MouseDown</i>. The dragging of the mouse while mouse buttons and/or shifting keys (Ctrl, Alt, Shift) are pressed. When <i>action%</i> contains a mouse-drag value, <i>parameters\$</i> must contain <i>Coords=x1,y1,x2,y2</i>. <p>See Appendix E for a list of mouse click and drag values.</p> |
| <i>recMethod\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <i>HTMLId=\$</i>. The text from the ID attribute of the HTML object. ▶ <i>HTMLText=\$</i>. The visible text of a Web page SELECT form element. The text is from the Value attribute of the OPTION tag. ▶ <i>HTMLTitle=\$</i>. The text from the Title attribute of the HTML object. ▶ <i>ID=%</i>. The object's internal Windows ID. ▶ <i>Index=%</i>. The number of the object among all objects identified with the same base recognition method. Typically, <i>Index</i> is used after another recognition method qualifier — for example, <i>Name=\$;Index=%</i>. ▶ <i>Label=\$</i>. The text of the label object that immediately precedes the combo edit box in the internal order (Z order) of windows. |

▶ ▶ ▶



| Syntax Element | Description |
|----------------------|---|
| | <ul style="list-style-type: none"> ▶ <code>Name=\$</code>. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a <code>command</code> button might be <code>Command1</code>. ▶ <code>ObjectIndex=%</code>. The number of the object among all objects of the same type in the same window. ▶ <code>State=\$</code>. An optional qualifier for any other recognition method. There are two possible values for this setting: <code>Enabled</code> and <code>Disabled</code>. The default state is the state of the current context window (as set in the most recent <code>Window SetContext</code> command), or <code>Enabled</code> if the state has not been otherwise declared. ▶ <code>VisualText=\$</code>. An optional setting used to identify an object by its prior label. It is for user clarification only and does not affect object recognition. |
| <i>parameters</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <code>Coords=x,y</code>. If <i>action</i>% is a mouse click, specifies the coordinates of the click, relative to the top left of the object. ▶ <code>Coords=x1,y1,x2,y2</code>. If <i>action</i>% is a mouse drag, specifies the coordinates, where <i>x1,y1</i> are the starting coordinates of the drag, and <i>x2,y2</i> are the ending coordinates of the drag. The coordinates are relative to the top left of the object. |

Comments None.

Example This example clicks the second combo edit box in the window (`ObjectIndex=2`) at *x,y* coordinates of **59,10**.

```
ComboEditBox Click, "ObjectIndex=2", "Coords=59,10"
```

See Also `ComboBox` `EditBoxVP`
 `ComboBox` `ListBoxVP`

ComboEditBoxVP

Verification Point Command

Description Establishes a verification point for a combo edit box control.

Syntax `Result = ComboEditBoxVP (action%,recMethod$,parameters$)`

| Syntax Element | Description |
|--------------------|---|
| <i>action%</i> | <p>The type of verification to perform. Valid values:</p> <ul style="list-style-type: none"> ▶ <code>CompareNumeric</code>. Captures the numeric value of the text of the object and compares it to the value of <i>parameters\$</i> Value or Range. <i>parameters\$</i> VP and either Value or Range are required; ExpectedResult and Wait are optional. ▶ <code>CompareProperties</code>. Captures object properties information for the object and compares it to a recorded baseline. <i>parameters\$</i> VP is required; ExpectedResult and Wait are optional. ▶ <code>CompareText</code>. Captures the text of the object and compares it to a recorded baseline. <i>parameters\$</i> VP and Type are required; ExpectedResult and Wait are optional. ▶ <code>VerifyIsBlank</code>. Checks that the object has no text. <i>parameters\$</i> VP is required; ExpectedResult and Wait are optional. |
| <i>recMethod\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <code>HTMLId=\$</code>. The text from the ID attribute of the HTML object. ▶ <code>HTMLText=\$</code>. The visible text of a Web page SELECT form element. The text is from the Value attribute of the OPTION tag. ▶ <code>HTMLTitle=\$</code>. The text from the Title attribute of the HTML object. ▶ <code>ID=%</code>. The object's internal Windows ID. ▶ <code>Index=%</code>. The number of the object among all objects identified with the same base recognition method. Typically, Index is used after another recognition method qualifier — for example, <code>Name=\$; Index=%</code>. ▶ <code>Label=\$</code>. The text of the label object that immediately precedes the combo edit box in the internal order (Z order) of windows. |

▶ ▶ ▶

▶ ▶ ▶

| Syntax Element | Description |
|---------------------|---|
| | <ul style="list-style-type: none"> ▶ Name=\$. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be Command1. ▶ ObjectIndex=%. The number of the object among all objects of the same type in the same window. |
| <i>parameters\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ ExpectedResult=%. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> - PASS. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail. - FAIL. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail. ▶ Range=&, &. Used with the action CompareNumeric when a numeric range comparison is being performed, as in Range=2, 12 (test for numbers in this range). The values are inclusive. ▶ Type=\$. Specifies the verification method to use for CompareText actions. The possible values are: CaseSensitive, CaseInsensitive, FindSubStr, FindSubStrI (case insensitive), and UserDefined. See <i>Comments</i> for more information. If UserDefined is specified, two additional parameters are required: <ul style="list-style-type: none"> - DLL=\$. The full path and file name of the library that contains the function. - Function=\$. The name of the custom function to use in comparing the text. ▶ Value=&. Used with the action CompareNumeric when a numeric equivalence comparison is being performed, as in Value=25 (test against the value 25). ▶ VP=\$. The verification point ID. IDs must be unique within a script. Required for all verification points. ▶ Wait=%, %. A Wait State that specifies the verification point's Retry value and a Timeout value, as in Wait=10, 40 (retry the test every 10 seconds, but time out the test after 40 seconds). |

ComboBox

Comments This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

With the `Type=$` parameter, `CaseSensitive` and `CaseInsensitive` require a full match between the current baseline text and the text captured during playback. With `FindSubStr` and `FindSubStrI`, the current baseline can be a substring of the text captured during playback. The substring can appear anywhere in the playback text. To modify the current baseline text, double-click the verification point name in the Robot Asset pane (to the left of the script).

Example This example captures the text of the second combo edit box in the window (`ObjectIndex=2`) and performs a case-sensitive comparison with the recorded baseline in verification point `VPNESTED`. At playback, the comparison is retried every 3 seconds and times out after 30 seconds.

```
Result = ComboEditBoxVP (CompareText, "ObjectIndex=2",  
"VP=VPNESTED;Type=CaseSensitive;Wait=3,30")
```

See Also

| | |
|-------------------------|------------------------|
| <code>ComboBoxVP</code> | <code>EditBoxVP</code> |
| <code>ComboBoxVP</code> | <code>ListBoxVP</code> |

ComboBox

User Action Command

»»SQA»

Description Performs an action on a combo list box (the list box part of a combo box).

Syntax `ComboBox action%, recMethod$, parameters$`

| Syntax Element | Description |
|----------------------|--|
| <code>action%</code> | One of these actions: <ul style="list-style-type: none">▶ <code>MakeSelection</code>. Selects the specified item from a Java combo box. Used only for the Java environment. <code>recMethod\$</code> must contain one of the Java recognition methods, and <code>parameters\$</code> must contain either <code>Text</code> or <code>Index</code>.▶ <code>MouseClicked</code>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <code>action%</code> contains a mouse-click value, <code>parameters\$</code> must contain one of the following: <code>Text</code>, <code>ItemData</code>, <code>Index</code>, or <code>Coords=x,y</code>. |

▶ ▶ ▶

▶ ▶ ▶

| Syntax Element | Description | | | | | | | | | | |
|--------------------|--|-----------------|----------------|-------------|----------------|----------------|--------------|------------|--------------|-----------|-----------|
| | <ul style="list-style-type: none"> ▶ <i>MouseDown</i>. The dragging of the mouse while mouse buttons and/or shifting keys (Ctrl, Alt, Shift) are pressed. When <i>action%</i> contains a mouse-drag value, <i>parameters\$</i> must contain <i>Coords=x1,y1,x2,y2</i>. See Appendix E for a list of mouse click and drag values. ▶ <i>ScrollAction</i>. One of these scroll actions: <table border="0" style="margin-left: 20px;"> <tr> <td>ScrollPageRight</td> <td>ScrollPageDown</td> </tr> <tr> <td>ScrollRight</td> <td>ScrollLineDown</td> </tr> <tr> <td>ScrollPageLeft</td> <td>ScrollPageUp</td> </tr> <tr> <td>ScrollLeft</td> <td>ScrollLineUp</td> </tr> <tr> <td>HScrollTo</td> <td>VScrollTo</td> </tr> </table> <p>HScrollTo and VScrollTo take the required parameter <i>Position=%</i>.</p> <p>If Robot cannot interpret the action being applied to a scroll bar, which happens with certain custom standalone scroll bars, it records the action as a click or drag.</p> | ScrollPageRight | ScrollPageDown | ScrollRight | ScrollLineDown | ScrollPageLeft | ScrollPageUp | ScrollLeft | ScrollLineUp | HScrollTo | VScrollTo |
| ScrollPageRight | ScrollPageDown | | | | | | | | | | |
| ScrollRight | ScrollLineDown | | | | | | | | | | |
| ScrollPageLeft | ScrollPageUp | | | | | | | | | | |
| ScrollLeft | ScrollLineUp | | | | | | | | | | |
| HScrollTo | VScrollTo | | | | | | | | | | |
| <i>recMethod\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <i>HTMLId=\$</i>. The text from the ID attribute of the HTML object. ▶ <i>HTMLText=\$</i>. The text of an item in a Web page SELECT form element. The text is from the Value attribute of the OPTION tag. ▶ <i>HTMLTitle=\$</i>. The text from the Title attribute of the HTML object. ▶ <i>ID=%</i>. The object's internal Windows ID. ▶ <i>Index=%</i>. The number of the object among all objects identified with the same base recognition method. Typically, <i>Index</i> is used after another recognition method qualifier — for example, <i>Name=\$; Index=%</i>. ▶ <i>JavaText=\$</i>. A label that identifies the object in the user interface. ▶ <i>Label=\$</i>. The text of the label object that immediately precedes the combo list box in the internal order (Z order) of windows. ▶ <i>Name=\$</i>. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be <i>Command1</i>. ▶ <i>ObjectIndex=%</i>. The number of the object among all objects of the same type in the same window. | | | | | | | | | | |

▶ ▶ ▶

ComboBox

▶ ▶ ▶

| Syntax Element | Description |
|---------------------|---|
| | <ul style="list-style-type: none"> ▶ <code>State=\$</code>. An optional qualifier for any other recognition method. There are two possible values for this setting: <code>Enabled</code> and <code>Disabled</code>. The default state is the state of the current context window (as set in the most recent <code>Window SetContext</code> command), or <code>Enabled</code> if the state has not been otherwise declared. ▶ <code>Type=\$</code>. An optional qualifier for recognition methods. Used to identify the object within a specific environment. ▶ <code>VisualText=\$</code>. An optional setting used to identify an object by its prior label. It is for user clarification only and does not affect object recognition. |
| <i>parameters\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <code>Coords=x,y</code>. If <i>action%</i> is a mouse click, specifies the coordinates of the click, relative to the top left of the object. Robot uses this parameter only if the item contents or index cannot be retrieved — for example, if the combo list box is empty or disabled. ▶ <code>Coords=x1,x2,y1,y2</code>. If <i>action%</i> is a mouse drag, specifies the coordinates, where <i>x1,y1</i> are the starting coordinates of the drag, and <i>x2,y2</i> are the ending coordinates. The coordinates are relative to the top left of the object. ▶ <code>Index=%</code>. If <i>action%</i> is a mouse click or <code>MakeSelection</code>, identifies the index of an item in the list. ▶ <code>ItemData=&</code>. If <i>action%</i> is a mouse click, identifies the internal value, or <code>ItemData</code>, associated with an item in the list. All items in a list have an associated value. The uniqueness and significance of this value is entirely up to the application. Robot uses this parameter only if the combo list box item's text cannot be retrieved (for example, if it is an <code>OwnerDrawn</code> combo box), and if the <code>Identify List Selections By recording</code> option is set to <code>Contents</code>. ▶ <code>Position=%</code>. If <i>action%</i> is a <code>VScrollTo</code> or <code>HScrollTo</code>, specifies the scroll bar value of the new scrolled-to position. Every scroll bar has an internal range, and this value is specific to that range. ▶ <code>Text=\$</code>. If <i>action%</i> is a mouse click or <code>MakeSelection</code>, identifies the text of an item in the list. |

Comments None.

Example This example clicks the item identified by the text VGA in the combo list box identified by the label Display.

```
ComboBox Click, "Label=Display:", "Text=VGA"
```

This example clicks a select list item with a Value attribute of 1. The list is located within the Web page frame named Main.

```
ComboBox Click,
"Type=HTMLFrame;HTMLId=Main;\;Type=ComboBox;
Name=Selectlist", "Index=1"
```

See Also ComboBox EditBox
 ComboEditBox ListBox

ComboBoxVP

Verification Point Command



Description Establishes a verification point for a combo list box.

Syntax *Result* = **ComboBoxVP** (*action%*, *recMethod\$*, *parameters\$*)

| Syntax Element | Description |
|----------------|---|
| <i>action%</i> | <p>The type of verification to perform. Valid values:</p> <ul style="list-style-type: none"> ▶ Compare. Captures the entire textual contents of the object into a grid and compares it to a recorded baseline. <i>parameters\$ VP</i> is required; <i>ExpectedResult</i> and <i>Wait</i> are optional. ▶ CompareNumeric. Captures the numeric value of the text of the object and compares it to the value of <i>parameters\$ Value</i> or <i>Range</i>. <i>parameters\$ VP</i> and either <i>Value</i> or <i>Range</i> are required; <i>ExpectedResult</i> and <i>Wait</i> are optional. ▶ CompareProperties. Captures object properties information for the object and compares it to a recorded baseline. <i>parameters\$ VP</i> is required; <i>ExpectedResult</i> and <i>Wait</i> are optional. ▶ CompareText. Captures the text of the currently selected item and compares it to a recorded baseline. <i>parameters\$ VP</i> and <i>Type</i> are required; <i>ExpectedResult</i> and <i>Wait</i> are optional. |



ComboBoxVP

▶ ▶ ▶

| Syntax Element | Description |
|----------------------|--|
| <i>recMethod</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ HTMLId=\$. The text from the ID attribute of the HTML object. ▶ HTMLText=\$. The text of an item in a Web page SELECT form element. The text is from the Value attribute of the OPTION tag. ▶ HTMLTitle=\$. The text from the Title attribute of the HTML object. ▶ ID=%. The object's internal Windows ID. ▶ Index=%. The number of the object among all objects identified with the same base recognition method. Typically, Index is used after another recognition method qualifier — for example, Name=\$; Index=%. ▶ JavaText=\$. A label that identifies the object in the user interface. ▶ Label=\$. The text of the label object that immediately precedes the combo list box in the internal order (Z order) of windows. ▶ Name=\$. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be Command1. ▶ ObjectIndex=%. The number of the object among all objects of the same type in the same window. |
| <i>parameters</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ ExpectedResult=%. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> – PASS. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail. – FAIL. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail. ▶ Range=&, &. Used with the action CompareNumeric when a numeric range comparison is being performed, as in Range=2, 12 (test for numbers in this range). The values are inclusive. |

▶ ▶ ▶



| Syntax Element | Description |
|----------------|---|
| | <ul style="list-style-type: none"> ▶ Type=\$. Specifies the verification method to use for CompareText actions. The possible values are: CaseSensitive, CaseInsensitive, FindSubStr, FindSubStrI (case insensitive), and UserDefined. See <i>Comments</i> for more information. If UserDefined is specified, two additional parameters are required: <ul style="list-style-type: none"> - DLL=\$. The full path and file name of the library that contains the function. - Function=\$. The name of the custom function to use in comparing the text. ▶ Value=&. Used with the action CompareNumeric when a numeric equivalence comparison is being performed, as in Value=25 (test against the value 25). ▶ VP=\$. The verification point ID. IDs must be unique within a script. Required for all verification points. ▶ Wait=% , %. A Wait State that specifies the verification point's Retry value and a Timeout value, as in Wait=10 , 40 (retry the test every 10 seconds, but time out the test after 40 seconds). |

Comments This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

With the Type=\$ parameter, CaseSensitive and CaseInsensitive require a full match between the current baseline text and the text captured during playback. With FindSubStr and FindSubStrI, the current baseline can be a substring of the text captured during playback. The substring can appear anywhere in the playback text. To modify the current baseline text, double-click the verification point name in the Robot Asset pane (to the left of the script).

Example This example captures the properties of the combo list box control identified by the label Display and compares them to the recorded baseline in verification point VPNEW. At playback, the comparison is retried every 2 seconds and times out after 30 seconds.

```
Result = ComboBoxVP (CompareProperties,
    "Label=Display:", "VP=VPNEW;Wait=2,30")
```

See Also ComboBox EditBox
 ComboEditBox ListBox

Command

Command Function

Description Returns the command line specified when the MAIN sub procedure was invoked.

Syntax **Command**[\$]

| Syntax Element | Description |
|----------------|--|
| \$ | Optional. If specified, the return type is String. If omitted, the function returns a Variant of VarType 8 (String). |

Comments After the MAIN sub procedure returns, further calls to the Command function will yield an empty string. This function might not be supported in some implementations of SQABasic.

Example This example opens the file entered by the user on the command line.

```
Sub main
  Dim filename as String
  Dim cmdline as String
  Dim cmdlength as Integer
  Dim position as Integer
  cmdline=Command
  If cmdline="" then
    MsgBox "No command line information."
    Exit Sub
  End If
  cmdlength=Len(cmdline)
  position=InStr(cmdline,Chr(32))
  filename=Mid(cmdline,position+1,cmdlength-position)
  On Error Resume Next
  Open filename for Input as #1
  If Err<>0 then
    MsgBox "Error loading file."
    Exit Sub
  End If
  MsgBox "File " & filename & " opened."
  Close #1
  MsgBox "File " & filename & " closed."
End Sub
```

See Also AppActivate InputKeys
DoEvents Shell
Environ

Const Statement

Description Declares symbolic constants for use in an SQABasic program.

Syntax `[Global] Const constantName [As type]= expression
[,constantName [As type]= expression]...`

| Syntax Element | Description |
|---------------------|---|
| <i>constantName</i> | The variable name to contain a constant value. |
| <i>type</i> | The data type of the constant (Number or String). |
| <i>expression</i> | Any expression that evaluates to a constant number. |

Comments Instead of using the *As* clause, the type of the constant can be specified by using a type-declaration character as a suffix (# for numbers, \$ for strings) to the *constantName*. If no type-declaration character is specified, the type of the *constantName* is derived from the type of the expression.

If *Global* is specified, the constant is validated at module load time. If the constant has already been added to the runtime global area, the constant's type and value are compared to the previous definition, and the load fails if a mismatch is found. This is useful as a mechanism for detecting version mismatches between modules.

Example This example divides the US national debt by the number of people in the country to find the amount of money each person would have to pay to wipe it out. This figure is converted to a Long integer and formatted as *Currency*.

```
Sub main
  Dim debt As Single
  Dim msgtext
  Const Populace = 250000000
  debt=InputBox("Enter the current US national debt:")
  msgtext = "The debt per citizen is: "
  msgtext = msgtext + Format(CLng(Debt/Populace), "Currency")
  MsgBox msgtext
End Sub
```

See Also `Declare` `Let`
`Deftype` `Type`
`Dim`

Cos

Cos Function

Description Returns the cosine of an angle.

Syntax **Cos** (*number*)

| Syntax Element | Description |
|----------------|----------------------|
| <i>number</i> | An angle in radians. |

Comments The return value will be between -1 and 1. The return value is a single-precision number if the angle has a data type Integer, Currency, or is a single-precision value. The return value will be a double precision value if the angle has a data type Long, Variant, or is a double-precision value.

The angle can be either positive or negative. To convert degrees to radians, multiply by (PI/180). The value of PI is approximately 3.14159.

Example This example finds the length of a roof, given its pitch and the distance of the house from its center to the outside wall.

```
Sub Main
  Dim bwidth, roof, pitch
  Dim msgtext
  Const PI=3.14159
  Const conversion=PI/180
  pitch=InputBox("Enter roof pitch in degrees")
  pitch=Cos(pitch*conversion)
  bwidth=InputBox("Enter 1/2 of house width in feet")
  roof=bwidth/pitch
  msgtext = "The length of the roof is "
  msgtext = msgtext + Format(roof, "##.##") & " feet."
  MsgBox msgtext
End Sub
```

See Also Atn
Sin
Tan
Derived Trigonometric Functions (Appendix D)

CreateObject

Function

Description Creates a new OLE2 automation object.

Syntax **CreateObject**(*class*)

| Syntax Element | Description |
|----------------|---|
| <i>class</i> | The name of the application, a period, and the name of the object to be used. |

Comments To create an object, you first must declare an object variable, using `Dim`, and then set the variable equal to the new object, as follows:

```
Dim OLE2 As Object
Set OLE2 = CreateObject("spoly.cpoly")
```

To refer to a method or property of the newly created object, use the syntax *objectvar.property* or *objectvar.method*, as follows:

```
OLE2.reset
```

Refer to the documentation provided with your OLE2 automation server application for correct application and object names.

Example This example uses the `CreateObject` function to open the software product VISIO (if it is not already open).

```
Sub main
  Dim visio As Object
  Dim doc As Object
  Dim i As Integer, doccount As Integer

  'Initialize Visio
  on error resume next
  Set visio = GetObject(,"visio.application")
  If (visio Is Nothing) then
    Set visio = CreateObject("visio.application")
    If (visio Is Nothing) then
      MsgBox "Couldn't find Visio!"
    Exit Sub
  End If
End If
MsgBox "Visio is open."
End Sub
```

See Also Class List Nothing
 GetObject Object Class
 Is Typeof
 New

CSng

CSng Function

Description Converts an expression to the data type `Single`.

Syntax **CSng**(*expression*)

| Syntax Element | Description |
|-------------------|---|
| <i>expression</i> | Any expression that can evaluate to a number. |

Comments The *expression* must have a value within the range allowed for the `Single` data type, or an error occurs.

Strings that cannot be converted to an integer result in a `Type Mismatch` error. Variants containing null result in an `Illegal Use of Null` error.

Example This example calculates the factorial of a number. A factorial (represented as an exclamation mark, !) is the product of a number and each integer between it and the number 1. For example, 5 factorial, or 5!, is the product of 5*4*3*2*1, or the value 120.

```
Sub main
  Dim number as Integer
  Dim factorial as Double
  Dim x as Integer
  Dim msgtext
  number=InputBox("Enter an integer between 1 and 170:")
  If number<=0 then
    Exit Sub
  End If
  factorial=1
  For x=number to 2 step -1
    factorial=factorial*x
  Next x
  Rem If number =<35, then its factorial is small enough to
  Rem be stored as a single-precision number
  If number<35 then
    factorial=CSng(factorial)
  End If
  msgtext="The factorial of " & number & " is: " & factorial
  MsgBox msgtext
End Sub
```

See Also CCur CStr
 CDBl CVar
 CInt CDate
 CLng

CStr

Function

Description Converts an expression to the data type `String`.

Syntax `CStr(expression)`

| Syntax Element | Description |
|-------------------|---|
| <i>expression</i> | <p>Any expression that can evaluate to a number. The <code>CStr</code> statement accepts any type of <i>expression</i>:</p> <ul style="list-style-type: none"> ▶ <code>Boolean</code>. A <code>String</code> containing <code>TRUE</code> or <code>FALSE</code>. ▶ <code>Date</code>. A <code>String</code> containing a date. ▶ <code>Empty</code>. A zero-length <code>String</code> (" "). ▶ <code>Error</code>. A <code>String</code> containing the word <code>Error</code> followed by the error number. ▶ <code>Null</code>. A runtime error. ▶ <code>Other Numeric</code>. A <code>String</code> containing the number. |

Comments None.

Example This example converts a variable from a value to a string and displays the result. Variant type 5 is `Double` and type 8 is `String`.

```
Sub main
    Dim var1
    Dim msgtext as String
    var1=InputBox("Enter a number:")
    var1=var1+10
    msgtext="Your number plus 10 is: " & var1 & Chr(10)
    msgtext=msgtext & "making its Variant type: " & Vartype(var1)
    MsgBox msgtext
    var1=CStr(var1)
    msgtext="After conversion to a string," & Chr(10)
    msgtext=msgtext & "the Variant type is: " & Vartype(var1)
    MsgBox msgtext
End Sub
```

See Also

| | | |
|-------------------|-------------------|---------------------|
| <code>Asc</code> | <code>CInt</code> | <code>CVar</code> |
| <code>CCur</code> | <code>CLng</code> | <code>CVDate</code> |
| <code>Cdbl</code> | <code>CSng</code> | <code>Format</code> |
| <code>Chr</code> | | |

'\$CStrings

'\$CStrings Metacommand

»»SQA»

Description Tells the compiler to treat a backslash character (\) inside a string as an escape character.

Syntax `'$CStrings [Save | Restore]`

| Syntax Element | Description |
|----------------|---|
| Save | Saves the current '\$CStrings setting. |
| Restore | Restores a previously saved \$CStrings setting. |

Comments This treatment of a backslash in a string is based on the C language.

All metacommands must begin with an apostrophe (') and are recognized by the compiler only if the command starts at the beginning of a line.

Save and Restore operate as a stack and allow the user to change the setting for a range of the program without impacting the rest of the program.

The supported special characters are:

| | | | |
|----|--------------------|----|----------------|
| \n | Newline (Linefeed) | \f | Formfeed |
| \t | Horizontal Tab | \\ | Backslash |
| \v | Vertical Tab | \' | Single Quote |
| \b | Backspace | \" | Double Quote |
| \r | Carriage Return | \0 | Null Character |

The instruction "Hello\r World" is the equivalent of "Hello" + Chr\$(13) + "World".

In addition, any character can be represented as a 3-digit octal code or a 3-digit hexadecimal code:

`\ddd` Octal Code `\xddd` Hexadecimal Code

For both hexadecimal and octal, fewer than 3 characters can be used to specify the code as long as the subsequent character is not a valid (hex or octal) character.

To tell the compiler to return to the default string processing mode, where the backslash character has no special meaning, use the '\$NoCStrings metacommand.

Example This example displays two lines, the first time using the C-language characters `\n` for a carriage return and line feed.

```
Sub main
  '$CStrings
  MsgBox "This is line 1\n This is line 2 (using C Strings)"
  '$NoCStrings
  MsgBox "This is line 1" +Chr$(13)+Chr$(10)+"This is
    line 2 (using Chr)"
End Sub
```

See Also `$Include` `Rem`
`$NoCStrings`

CurDir

Function

Description Returns the default directory (and drive) for the specified drive.

Syntax `CurDir[$] [(drive$)]`

| Syntax Element | Description |
|----------------------|--|
| <code>\$</code> | Optional. If specified, the return type is <code>String</code> . If omitted, the function will return a <code>Variant of VarType 8 (String)</code> . |
| <code>drive\$</code> | A string expression containing the drive to search. |

Comments The drive must exist, and must be within the range specified in the `LASTDRIVE` statement of the `CONFIG.SYS` file. If a null argument (" ") is supplied, or if no `drive$` is indicated, the path for the default drive is returned.

To change the current drive, use `ChDrive`. To change the current directory, use `ChDir`.

Example This example changes the current directory to `C:\WINDOWS`, if it is not already the default.

```
Sub main
  Dim newdir as String
  newdir="c:\windows"
  If CurDir <> newdir then
    ChDir newdir
  End If
  MsgBox "The default directory is now: " & newdir
End Sub
```

See Also `ChDir` `MkDir`
`ChDrive` `Rmdir`
`Dir`

CVar

CVar

Function

Description Converts an expression to the data type Variant.

Syntax **CVar**(*expression*)

| Syntax Element | Description |
|-------------------|---|
| <i>expression</i> | Any expression that can evaluate to a number. |

Comments CVar accepts any type of *expression*.
CVar generates the same result as you would get by assigning the *expression* to a Variant variable.

Example This example converts a string variable to a variant variable.

```
Sub main
  Dim answer as Single
  answer=100.5
  MsgBox "'Answer' is DIMed as Single with the value: " & answer
  answer=CVar(answer)
  answer=Fix(answer)
  MsgBox "'Answer' is now a variant with type: " & VarType(answer)
End Sub
```

See Also CCur CLng CStr
 CDb1 CSng CVDate
 CInt

CVDate

Function

Description Converts an expression to the data type Variant Date.

Syntax **CVDate**(*expression*)

| Syntax Element | Description |
|-------------------|---|
| <i>expression</i> | Any expression that can evaluate to a number. |

Comments CVDate accepts both string and numeric values.

The CVDate function returns a Variant of VarType 7 (date) that represents a date from January 1, 100 through December 31, 9999. A value of 1 represents December 31, 1899, and a value of -1 represents December 29, 1899. Times are represented as fractional days.

With this function, a two-digit year is converted to a four-digit year, as follows:

- ▶ 00 through 29 is converted to 2000 through 2029
- ▶ 30 through 99 is converted to 1930 through 1999

When exchanging data information with external data sources or external programs, you should use double-precision floating point numbers or data strings with at least four characters for identifying the century.

Example This example displays the date for one week from the date entered by the user.

```
Sub main
Dim str1 as String
Dim x as Integer
Dim nextweek
Dim msgtext
i: str1=InputBox$("Enter a date:")
answer=IsDate(str1)
If answer=-1 then
str1=CVDate(str1)
nextweek=DateValue(str1)+7
msgtext="One week from the date entered is:"
msgtext=msgtext & Format(nextweek,"dddddd")
MsgBox msgtext
Else
MsgBox "Invalid date or format. Try again."
Goto i
End If
End Sub
```

| | | | |
|----------|------|------|--------|
| See Also | Asc | CInt | CVar |
| | CCur | CLng | Format |
| | Cdbl | CSng | Val |
| | Chr | CStr | |

DataWindow

User Action Command

»»»SQA»

Description Performs an action on a PowerBuilder DataWindow.

Syntax **DataWindow** *action%*, *recMethod\$*, *parameters\$*

| Syntax Element | Description |
|--------------------|---|
| <i>action%</i> | <p>One of these actions:</p> <ul style="list-style-type: none"> ▶ <i>MouseClick</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). ▶ When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> identifies the DataWindow <i>row</i> that was clicked. See <i>Comments</i> for more information. ▶ <i>MouseDown</i>. The dragging of the mouse while mouse buttons and/or shifting keys (Ctrl, Alt, Shift) are pressed. When <i>action%</i> contains a mouse-drag value, <i>parameters\$</i> must contain <i>Coords=x1,y1,x2,y2</i>. See Appendix E for a list of mouse click and drag values. ▶ <i>ScrollAction</i>. One of these scroll actions: <ul style="list-style-type: none"> ScrollPageRight ScrollPageDown ScrollRight ScrollLineDown ScrollPageLeft ScrollPageUp ScrollLeft ScrollLineUp HScrollTo VScrollTo HScrollTo and VScrollTo take the required parameter <i>Position=%</i>. |
| <i>recMethod\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <i>ID=%</i>. The object's internal Windows ID. ▶ <i>Name=\$</i>. A name that a developer assigns to an object to uniquely identify the object in the development environment. Applies to both the DataWindow itself and to its child objects (such as columns). For example a clicked column might be identified as follows: <pre>"Name=datawindow;\;Name=col_custid"</pre> ▶ <i>ObjectIndex=%</i>. The number of the object among all objects of the same type in the same window. |

▶ ▶ ▶

▶ ▶ ▶

| Syntax Element | Description |
|---------------------|---|
| | <ul style="list-style-type: none"> ▶ <code>State=\$</code>. An optional qualifier for any other recognition method. There are two possible values for this setting: <code>Enabled</code> and <code>Disabled</code>. The default state is the state of the current context window (as set in the most recent <code>Window SetContext</code> command), or <code>Enabled</code> if the state has not been otherwise declared. ▶ <code>Text=\$</code>. The caption of the DataWindow. Wildcards are not supported. ▶ <code>VisualText=\$</code>. An optional setting used to identify an object by its visible text. It is for user clarification only and does not affect object recognition. |
| <i>parameters\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <code>Col=%;Value=x</code>. If <i>action%</i> is a mouse click, these two parameters specify the row being clicked: <ul style="list-style-type: none"> – <code>Col</code> is the numeric position of a column in the DataWindow (the leftmost column = 1, the next column = 2, and so forth) – <code>Value</code> is the contents of the cell located at the intersection of column <code>Col</code> and the clicked row ▶ <code>ColName=\$;Value=x</code>. If <i>action%</i> is a mouse click, these two parameters specify the row being clicked: <ul style="list-style-type: none"> – <code>ColName</code> is the developer-assigned object name of a column in the DataWindow – <code>Value</code> is the contents of the cell located at the intersection of column <code>ColName</code> and the clicked row ▶ <code>Coords=x,y</code>. If <i>action%</i> is a mouse click, specifies the <i>x,y</i> coordinates of the click, relative to the top left of the cell or column being clicked. ▶ <code>Coords=x1,y1,x2,y2</code>. If <i>action%</i> is a mouse drag, specifies the coordinates, where <i>x1,y1</i> are the starting coordinates of the drag, and <i>x2,y2</i> are the ending coordinates. The coordinates are relative to the top left of the object or the item. ▶ <code>CurrentRow</code>. If <i>action%</i> is a mouse click, the currently selected row in the DataWindow is clicked. ▶ <code>Index=%</code>. An optional 1-based clarifier for <code>Text=\$</code> or column/value <i>parameters\$</i>. For example, <code>Text=Yes;Index=3</code> specifies the third DataWindow row containing the value <code>Yes</code>. ▶ <code>LastRow</code>. If <i>action%</i> is a mouse click, the last row in the DataWindow is clicked. |

▶ ▶ ▶

DataWindow

▶ ▶ ▶

| Syntax Element | Description |
|----------------|---|
| | <ul style="list-style-type: none"> ▶ <code>Position=%</code>. If <code>action%</code> is <code>VScrollTo</code> or <code>HScrollTo</code>, specifies the scroll bar value of the new scrolled-to position in the scroll box. Every scroll bar has an internal range, and this value is specific to that range. ▶ <code>Row=%</code>. If <code>action%</code> is a mouse click, the number of the DataWindow row being clicked (first row = 1). ▶ <code>Text=\$</code>. If <code>action%</code> is a mouse click, the visible text in the row being clicked. ▶ <code>VisibleRow=%</code>. If <code>action%</code> is a mouse click, the number of the visible row being clicked. The range of row numbers begins with the first visible row (first visible row = 1). |

Comments With mouse-click actions, `recMethod$` specifies the column being clicked, and `parameters$` specifies the row being clicked.

Whenever possible during recording, Robot specifies the clicked row by using one of these `parameters$` values (or pairs of values) in the following default order of priority:

1. `CurrentRow` (when the user action takes place in the currently selected row, and the user's previous action also took place in that row).
2. One or more pairs of a column identifier (`Col=%` or `ColName=$`) followed by `Value=x`. Robot uses as many column/value pairs as necessary to uniquely identify the clicked row — for example:

```
ColName=acct_type;Value=Savings;ColName=acct_number;Value=388217
```
3. `Text=$` (when the DataWindow is not editable and has less than four visible columns).
4. `Row=%`. If the current user action is in the last row of the DataWindow, and the action occurs in an editable column, Robot uses `LastRow`.
5. `Coords=x,y`.

Note the following points about column/value pairs:

- ▶ Value must *immediately* follow `Col` or `ColName`.
- ▶ The values are separated by a semicolon (;) — for example:

```
"ColName=custid;Value=0253319"
```

- ▶ The column identifier (`Col` or `ColName`) isn't necessarily the column that was clicked. Robot looks for one or more *key* columns of unique values. If a key column is found:
 - The column identifier specifies the key column
 - `Value` specifies the contents of the cell at the intersection of the key column and the row that the user clicked
- ▶ If there are no key columns, Robot uses as many column/value pairs as necessary to uniquely identify the clicked row, starting with the leftmost column.
- ▶ `parameters$` has a maximum length of 968 characters. If multiple column/row pairs cause `parameters$` to exceed the maximum length, Robot uses another way to uniquely identify the clicked row.

Robot treats the following pairs of `parameters$` values equally:

```
Row=0 and CurrentRow
Row=-1 and LastRow
```

Example

This example clicks the DataWindow cell that's identified by the column `custname` and the row specified by the column/value pair `Col=1;Value=11739`.

```
DataWindow Click, "Name=dw;\;Name=custname", "Col=1;Value=11739"
```

This example uses the relative row indicator `CurrentRow` and the coordinates of the click to specify the row being clicked.

```
DataWindow Click, "Name=dw;\;Name=custname", "CurrentRow;Coords=5,5"
```

This example uses the relative row indicator `VisibleRow=%` to specify that the second visible row is being clicked. Note that the clicked row may or may not be the second row in the entire DataWindow table.

```
DataWindow Click, "Name=dw;\;Name=custname", "VisibleRow=2"
```

See Also

DataWindowVP

DataWindowVP

Verification Point Command

Description Establishes a verification point for a PowerBuilder DataWindow.

Syntax `Result = DataWindowVP (action%, recMethod$, parameters$)`

| Syntax Element | Description |
|--------------------------|--|
| <code>action%</code> | <p>The type of verification to perform. Valid values:</p> <ul style="list-style-type: none"> ▶ <code>Compare</code>. Captures the data of the object into a grid and compares it to a recorded baseline. <code>parameters\$ VP</code> is required; <code>ExpectedResult</code> and <code>Wait</code> are optional. ▶ <code>CompareNumeric</code>. Captures the numeric value of the text of the object and compares it to the value of <code>parameters\$ Value or Range</code>. <code>parameters\$ VP</code> and either <code>Value or Range</code> are required; <code>ExpectedResult</code> and <code>Wait</code> are optional. ▶ <code>CompareProperties</code>. Captures object properties information for the object and compares it to a recorded baseline. <code>parameters\$ VP</code> is required; <code>ExpectedResult</code> and <code>Wait</code> are optional. ▶ <code>CompareText</code>. Captures the text of the caption of the DataWindow and compares it to a recorded baseline. <code>parameters\$ VP</code> and <code>Type</code> are required; <code>ExpectedResult</code> and <code>Wait</code> are optional. |
| <code>recMethod\$</code> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <code>ID=%</code>. The object's internal Windows ID. ▶ <code>Name=\$</code>. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be <code>Command1</code>. ▶ <code>ObjectIndex=%</code>. The number of the object among all objects of the same type in the same window. ▶ <code>Text=\$</code>. The caption of the DataWindow. Wildcards are not supported. |

» » »



| Syntax Element | Description |
|---------------------------|---|
| <code>parameters\$</code> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <code>ExpectedResult=%</code>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> - <code>PASS</code>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail. - <code>FAIL</code>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail. ▶ <code>Range=& , &</code>. Used with the action <code>CompareNumeric</code> when a numeric range comparison is being performed, as in <code>Range=2 , 12</code> (test for numbers in this range). The values are inclusive. ▶ <code>Type=\$</code>. Specifies the verification method to use for <code>CompareText</code> actions. The possible values are: <code>CaseSensitive</code>, <code>CaseInsensitive</code>, <code>FindSubStr</code>, <code>FindSubStrI</code> (case insensitive), and <code>UserDefined</code>. See <i>Comments</i> for more information. If <code>UserDefined</code> is specified, two additional parameters are required: <ul style="list-style-type: none"> - <code>DLL=\$</code>. The full path and file name of the library that contains the function. - <code>Function=\$</code>. The name of the custom function to use in comparing the text. ▶ <code>Value=&</code>. Used with the action <code>CompareNumeric</code> when a numeric equivalence comparison is being performed, as in <code>Value=25</code> (test against the value 25). ▶ <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points. ▶ <code>Wait=% , %</code>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10 , 40</code> (retry the test every 10 seconds, but time out the test after 40 seconds). |

Comments This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

Date (Function)

With the `Type=$` parameter, `CaseSensitive` and `CaseInsensitive` require a full match between the current baseline text and the text captured during playback. With `FindSubStr` and `FindSubStrI`, the current baseline can be a substring of the text captured during playback. The substring can appear anywhere in the playback text. To modify the current baseline text, double-click the verification point name in the Robot Asset pane (to the left of the script).

Example This example captures the contents of the `DataWindow` control identified by the PowerBuilder object name `dw_trans` and compares it to a recorded baseline in verification point `QBDW1`.

```
Result = DataWindowVP (Compare, "Name=dw_trans", "VP=QBDW1")
```

See Also None.

Date Function

Description Returns a string representing the current date.

Syntax `Date`[\$]

| Syntax Element | Description |
|----------------|---|
| \$ | Optional. If specified, the return type is <code>String</code> . If omitted, the function will return a <code>Variant</code> of <code>VarType 8 (String)</code> . |

Comments The `Date` function returns a ten character string.

Example This example displays the date for one week from the today's date (the current date on the computer).

```
Sub main
  Dim nextweek
  nextweek=CVar(Date)+7
  MsgBox "One week from today is: " & Format(nextweek, "dddd")
End Sub
```

See Also

| | |
|-----------------------------|-------------------------|
| <code>CVDate</code> | Time function |
| <code>Date</code> statement | Time statement |
| <code>Format</code> | Timer |
| <code>Now</code> | <code>TimeSerial</code> |

Date Statement

Description Sets the system date.

Syntax **Date** = *expression*

| Syntax Element | Description |
|-------------------|--|
| <i>Expression</i> | <p>A string in one of the following forms:</p> <p><i>mm-dd-yy</i> <i>mm-dd-yyyy</i> <i>mm/dd/yy</i> <i>mm/dd/yyyy</i></p> <p>where <i>mm</i> denotes a month (01-12), <i>dd</i> denotes a day (01-31), and <i>yy</i> or <i>yyyy</i> denotes a year (1980-2099).</p> |

Comments If *expression* is not already a Variant of VarType 7 (date), Date attempts to convert it to a valid date from January 1, 1980 through December 31, 2099. Date uses the Short Date format in the International section of Windows Control Panel to recognize day, month, and year if a string contains three numbers delimited by valid date separators. In addition, Date recognizes month names in either full or abbreviated form.

With this function, a two-digit year is converted to a four-digit year, as follows:

- ▶ 80 through 99 is converted to 1980 through 1999
- ▶ 00 through 79 is converted to 2000 through 2079

When exchanging data information with external data sources or external programs, you should use double-precision floating point numbers or data strings with at least four characters for identifying the century.

Example This example changes the system date to a date entered by the user.

```
Sub main
    Dim userdate
    Dim answer
    i: userdate= InputBox("Enter date for the system clock:")
    If userdate="" then
        Exit Sub
    End If
    answer=IsDate(userdate)
    If answer=-1 then
        Date=userdate
    Else
        MsgBox "Invalid date or format. Try again."
        Goto i
    End If
End Sub
```

DateSerial

See Also Date function
 Time function
 Time statement

DateSerial

Function

Description Returns a date value for year, month, and day specified.

Syntax **DateSerial**(*year%*, *month%*, *day%*)

| Syntax Element | Description |
|----------------|---|
| <i>year%</i> | A year between 100 and 9999, or a numeric expression. |
| <i>month%</i> | A month between 1 and 12, or a numeric expression. |
| <i>day%</i> | A day between 1 and 31, or a numeric expression. |

Comments The `DateSerial` function returns a `Variant` of `VarType 7` (date) that represents a date from January 1, 100 through December 31, 9999.

A numeric expression can be used for any of the arguments to specify a relative date: a number of days, months, or years before or after a certain date.

Example This example finds the day of the week New Year's day will be for the year 2000.

```
Sub main
  Dim newyearsday
  Dim daynumber
  Dim msgtext
  Dim newday as Variant
  Const newyear=2000
  Const newmonth=1
  Let newday=1
  newyearsday=DateSerial(newyear, newmonth, newday)
  daynumber=Weekday(newyearsday)
  msgtext="New Year's day 2000 is a " & Format(daynumber, "dddd")
  MsgBox msgtext
End Sub
```

See Also DateValue Now Weekday
 Day TimeSerial Year
 Month TimeValue

DateTime

User Action Command



Description Performs an action on a date and time picker (DTP) control.

Syntax `DateTime action%, recMethod$, parameters$`

| Syntax Element | Description |
|---------------------------|---|
| <code>action%</code> | <p>One of these mouse actions:</p> <ul style="list-style-type: none"> ▶ <i>MouseClicked</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <code>action%</code> contains a mouse-click value, <code>parameters\$</code> must contain <code>Coords=x,y</code>. ▶ <i>MouseDown</i>. The dragging of the mouse while mouse buttons and/or shifting keys (Ctrl, Alt, Shift) are pressed. When <code>action%</code> contains a mouse-drag value, <code>parameters\$</code> must contain <code>Coords=x1,y1,x2,y2</code>. <p>See Appendix E for a list of mouse click and drag values.</p> |
| <code>recMethod\$</code> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <code>ID=%</code>. The object's internal Windows ID. ▶ <code>Label=\$</code>. The text of the label object that immediately precedes the control in the Z order of windows. ▶ <code>Name=\$</code>. A unique name that a developer assigns to an object to identify the object in the development environment. For example, the object name for a command button might be <code>Command1</code>. ▶ <code>ObjectIndex=%</code>. The number of the object among all objects of the same type in the same window. ▶ <code>Text=\$</code>. The text displayed on the object. ▶ <code>VisualText=\$</code>. An optional setting used to identify an object by its prior label. It is for user clarification only and does not affect object recognition. |
| <code>parameters\$</code> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <code>Coords=x,y</code>. If <code>action%</code> is a mouse click, specifies the coordinates of the click, relative to the top left of the object. ▶ <code>Coords=x1,y1,x2,y2</code>. If <code>action%</code> is a mouse drag, specifies the coordinates, where <code>x1,y1</code> are the starting coordinates of the drag, and <code>x2,y2</code> are the ending coordinates. The coordinates are relative to the top left of the object. |

DateTimeVP

Comments None.

Example This example clicks the date and time picker control labeled “Select a Date” at *x,y* coordinates of 77,15.

```
DateTime Click, "Label=Select a Date", "Coords=77,15"
```

See Also Calendar
DateTimeVP

DateTimeVP

Verification Point Command

»»SQA»

Description Establishes a verification point for a date and time picker (DTP) control.

Syntax *Result* = **DateTimeVP** (*action%*, *recMethod\$*, *parameters\$*)

| Syntax Element | Description |
|--------------------|--|
| <i>action%</i> | The type of verification to perform. Valid value: <ul style="list-style-type: none">▶ <i>CompareProperties</i>. Captures object properties information for the object and compares it to a recorded baseline. <i>parameters\$ VP</i> is required; <i>ExpectedResult</i> and <i>Wait</i> are optional. |
| <i>recMethod\$</i> | Valid values: <ul style="list-style-type: none">▶ <i>ID=%</i>. The object's internal Windows ID.▶ <i>Label=\$</i>. The text of the label object that immediately precedes the control in the internal order (Z order) of windows.▶ <i>Name=\$</i>. A unique name that a developer assigns to an object to identify the object in the development environment. For example, the object name for a command button might be <i>Command1</i>.▶ <i>ObjectIndex=%</i>. The number of the object among all objects of the same type in the same window.▶ <i>Text=\$</i>. The text displayed on the object.▶ <i>VisualText=\$</i>. An optional setting used to identify an object by its prior label. It is for user clarification only and does not affect object recognition. |

▶ ▶ ▶



| Syntax Element | Description |
|----------------------|---|
| <i>parameters</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <code>ExpectedResult=%</code>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> – <code>PASS</code>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail. – <code>FAIL</code>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail. ▶ <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points. ▶ <code>Wait=%,%</code>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10,40</code> (retry the test every 10 seconds, but time out the test after 40 seconds). |

Comments This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

Example This example captures the properties of the date and time picker control labeled "Select a Date" and compares them to the recorded baseline in the verification point DATETIME1.

```
Result = DateTimeVP (CompareProperties, "Label=Select a Date",
"VP=DATETIME1")
```

See Also DateTime

DateValue

Function

Description Returns a date value for the string specified.

Syntax **DateValue**(*date*\$)

| Syntax Element | Description |
|----------------|-------------------------------------|
| <i>date</i> \$ | A string representing a valid date. |

Day

Comments The `DateValue` function returns a Variant of `VarType 7 (date)` that represents a date from January 1, 100 through December 31, 9999.

`DateValue` accepts several different string representations for a date. It makes use of the operating system's international settings for resolving purely numeric dates.

With this function, a two-digit year is converted to a four-digit year, as follows:

- ▶ 00 through 29 is converted to 2000 through 2029
- ▶ 30 through 99 is converted to 1930 through 1999

When exchanging data information with external data sources or external programs, you should use double-precision floating point numbers or data strings with at least four characters for identifying the century.

Example This example displays the date for one week from the date entered by the user.

```
Sub main
  Dim str1 as String
  Dim answer as Integer
  Dim nextweek
  Dim msgtext
i: str1=InputBox$("Enter a date:")
  answer=IsDate(str1)
  If answer=-1 then
    str1=CVDate(str1)
    nextweek=DateValue(str1)+7
    msgtext = "One week from your date is: "
    msgtext = msgtext + Format(nextweek,"dddddd")
    MsgBox msgtext
  Else
    MsgBox "Invalid date or format. Try again."
    Goto i
  End If
End Sub
```

See Also

| | | |
|-------------------------|-------------------------|----------------------|
| <code>DateSerial</code> | <code>Now</code> | <code>Weekday</code> |
| <code>Day</code> | <code>TimeSerial</code> | <code>Year</code> |
| <code>Month</code> | <code>TimeValue</code> | |

Day Function

Description Returns the day of the month (1-31) of a date-time value.

Syntax **Day**(*date*)

| Syntax Element | Description |
|----------------|---|
| <i>date</i> | Any expression that can evaluate to a date. |

Comments **Day** attempts to convert the input value of *date* to a date value. The return value is a Variant of VarType 2 (integer). If the value of *date* is null, a Variant of VarType 1 (null) is returned.

Example This example finds the month (1-12) and day (1-31) values for this Thursday.

```
Sub main
  Dim x, today, msgtext
  Today=DateValue(Now)
  Let x=0
  Do While Weekday(Today+x) <> 5
    x=x+1
  Loop
  msgtext="This Thursday is: " & Month(Today+x) & "/" & Day(Today+x)
  MsgBox msgtext
End Sub
```

See Also

| | | |
|----------------|--------|---------|
| Date function | Minute | Second |
| Date statement | Month | Weekday |
| Hour | Now | Year |

DDEAppReturnCode

Function

Description Returns a code received from an application on an open dynamic data exchange (DDE) channel.

Syntax **DDEAppReturnCode**()

Comments To open a DDE channel, use **DDEInitiate**. Use **DDEAppReturnCode** to check for error return codes from the server application after using **DDEExecute**, **DDEPoke** or **DDERequest**.

Example None.

See Also

| | |
|-------------|--------------|
| DDEExecute | DDERequest |
| DDEInitiate | DDETerminate |
| DDEPoke | |

DDEExecute

Statement

Description Sends one or more commands to an application via a dynamic-data exchange (DDE) channel.

Syntax **DDEExecute** *channel%*, *cmd\$*

| Syntax Element | Description |
|-----------------|---|
| <i>channel%</i> | An integer or expression for the channel number of the DDE conversation as returned by <code>DDEInitiate</code> . |
| <i>cmd\$</i> | One or more commands recognized by the application. |

Comments If *channel* does not correspond to an open channel, an error occurs.

You can also use the format described under `InputKeys` to send specific key sequences. If the server application cannot perform the specified command, an error occurs.

In many applications that support DDE, *cmd\$* can be one or more statements or functions in the application's macro language. Note that some applications require that each command received through a DDE channel be enclosed in brackets and quotation marks.

You can use a single `DDEExecute` instruction to send more than one command to an application.

Many commands require arguments in the form of strings enclosed in quotation marks. Because quotation marks indicate the beginning and end of a string in SQABasic, use `Chr$(34)` to include a quotation mark in a command string. For example, the following instruction tells Microsoft Excel to open MYFILE.XLS:

```
DDEExecute channelno, "[OPEN(" + Chr$(34) + "MYFILE.XLS" +
Chr$(34) + ")]"
```

Example

This example opens Microsoft Word, uses `DDEPoke` to write the text `Hello, World` to the open document (Untitled) and uses `DDEExecute` to save the text to the file TEMP001. The example assumes that WINWORD.EXE is in the path C:\MSOFFICE\WINWORD.

```
Sub main
  Dim channel as Integer
  Dim appname as String
  Dim topic as String
  Dim testtext as String
  Dim item as String
  Dim pcommand as String
  Dim msgtext as String
```

```

Dim answer as String
Dim x as Integer
Dim path as String
appname="WinWord"
path="c:\msoffice\winword\"
topic="System"
item="Page1"
testtext="Hello, world."
On Error Goto Errhandler
x=Shell(path & appname & ".EXE")
channel = DDEInitiate(appname, topic)
If channel=0 then
    MsgBox "Unable to open Word."
    Exit Sub
End If
DDEPoke channel, item, testtext
pcommand="[FileSaveAs .Name = "
pcommand=pcommand + Chr$(34) & "C:\TEMP001" & Chr$(34) & "]"
DDEExecute channel, pcommand
pcommand="[FileClose]"
DDEExecute channel, pcommand
msgtext="The text: " & testtext & " saved to C:\TEMP001."
msgtext=msgtext & Chr$(13) & "Delete? (Y/N)"
answer=InputBox(msgtext)
If answer="Y" or answer="y" then
    Kill "C:\TEMP001.doc"
End If
DDETerminate channel
Exit Sub
Errhandler:
If Err<>0 then
    MsgBox "DDE Access failed."
End If
End Sub

```

See Also DDEAppReturnCode DDETerminate
 DDEInitiate DDEPoke
 DDERequest

DDEInitiate

Function

Description Opens a dynamic-data exchange (DDE) channel and returns the DDE channel number (1,2, etc.).

Syntax **DDEInitiate**(*appname*\$, *topic*%)

| Syntax Element | Description |
|------------------|--|
| <i>appname</i> § | A string or expression for the name of the DDE application to talk to. |
| <i>topic</i> § | A string or expression for the name of a topic recognized by <i>appname</i> §. |

DDEInitiate

Comments If `DDEInitiate` is unable to open a channel, it returns zero (0).

Appname\$ is usually the name of the application's .EXE file without the .EXE file name extension. If the application is not running, `DDEInitiate` cannot open a channel and returns an error. Use `Shell` to start an application.

Topic\$ is usually an open file name. If *appname\$* does not recognize *topic\$*, `DDEInitiate` generates an error. Many applications that support DDE recognize a topic named `System`, which is always available and can be used to find out which other topics are available. For more information on the `System` topic, see `DDERequest`.

The maximum number of channels that can be open simultaneously is determined by the operating system and your system's memory and resources. If you aren't using an open channel, you should conserve resources by closing it using `DDETerminate`.

Example This example uses `DDEInitiate` to open a channel to Microsoft Word. It uses `DDERequest` to obtain a list of available topics (using the `System` topic). The example assumes that `WINWORD.EXE` is in the path `C:\MSOFFICE\WINWORD`.

```
Sub main
  Dim channel as Integer
  Dim appname as String
  Dim topic as String
  Dim item as String
  Dim msgtext as String
  Dim path as String
  appname="winword"
  topic="System"
  item="Topics"
  path="c:\msoffice\winword\"
  channel = -1
  x=Shell(path & appname & ".EXE")
  channel = DDEInitiate(appname, topic)
  If channel= -1 then
    msgtext="M/S Word not found -- please place on your path."
  Else
    On Error Resume Next
    msgtext="The Word topics available are:" & Chr$(13)
    msgtext=msgtext & Chr$(13) & DDERequest(channel,item)
    DDETerminate channel
    If Err<>0 then
      msgtext="DDE Access failed."
    End If
  End If
  MsgBox msgtext
End Sub
```

See Also

| | |
|-------------------------------|---------------------------|
| <code>DDEAppReturnCode</code> | <code>DDERequest</code> |
| <code>DDEExecute</code> | <code>DDETerminate</code> |
| <code>DDEPoke</code> | |

DDEPoke

Statement

Description Sends data to an application on an open dynamic-data exchange (DDE) channel.

Syntax **DDEPoke** *channel%*, *item\$*, *data\$*

| Syntax Element | Description |
|-----------------|---|
| <i>channel%</i> | An integer or expression for the open DDE channel number. |
| <i>item\$</i> | A string or expression for the name of an item in the currently opened topic. |
| <i>data\$</i> | A string or expression for the information to send to the topic. |

Comments If *channel%* does not correspond to an open channel, an error occurs.

When you open a channel to an application using `DDEInitiate`, you also specify a topic, such as a file name, to communicate with. The *item\$* is the part of the topic you want to send data to. `DDEPoke` sends data as a text string; you cannot send text in any other format, nor can you send graphics.

If the server application does not recognize *item\$*, an error occurs.

Example This example opens Microsoft Word, uses `DDEPoke` to write the text `Hello, World` to the open document (Untitled) and uses `DDEExecute` to save the text to the file `TEMP001`. The example assumes that `WINWORD.EXE` is in the path `C:\MSOFFICE\WINWORD`.

```
Sub main
    Dim channel as Integer
    Dim appname as String
    Dim topic as String
    Dim testtext as String
    Dim item as String
    Dim pcommand as String
    Dim msgtext as String
    Dim answer as String
    Dim x as Integer
    Dim path as String
    appname="WinWord"
    path="c:\msoffice\winword\"
    topic="System"
    item="Page1"
    testtext="Hello, world."
    On Error Goto Errhandler
    x=Shell(path & appname & ".EXE")
    channel = DDEInitiate(appname, topic)
```

DDERequest

```
If channel=0 then
    MsgBox "Unable to open Word."
    Exit Sub
End If
DDEPoke channel, item, testtext
pcommand="[FileSaveAs .Name = " & Chr$(34)
pcommand=pcommand & "C:\TEMP001" & Chr$(34) & "]"
DDEExecute channel, pcommand
pcommand="[FileClose]"
DDEExecute channel, pcommand
msgtext="The text " & testtext & " is saved to C:\TEMP001."
msgtext=msgtext & Chr$(13) & "Delete? (Y/N)"
answer=InputBox(msgtext)
If answer="Y" or answer="y" then
    Kill "C:\TEMP001.doc"
End If
DDETerminate channel
Exit Sub
Errhandler:
If Err<>0 then
    MsgBox "DDE Access failed."
End If
End Sub
```

See Also [DDEAppReturnCode](#) [DDERequest](#)
[DDEExecute](#) [DDETerminate](#)
[DDEInitiate](#)

DDERequest

Function

Description Returns data from an application through an open dynamic data exchange (DDE) channel.

Syntax **DDERequest**[\$] (*channel%*, *item%*)

| Syntax Element | Description |
|-----------------|--|
| <i>channel%</i> | An integer or expression for the open DDE channel number. |
| <i>item%</i> | A string or expression for the name of an item in the currently opened topic to get information about. Many applications that support DDE recognize a topic named <code>System</code> . Three standard items in the <code>System</code> topic are as follows: <ul style="list-style-type: none">▶ <code>SysItems</code>. A list of all items in the <code>System</code> topic▶ <code>Topics</code>. A list of available topics▶ <code>Formats</code>. A list of all the Clipboard formats supported |

Comments If *channel%* does not correspond to an open channel, an error occurs.

If the server application does not recognize *item\$*, an error occurs.

If `DDERequest` is unsuccessful, it returns an empty string ("").

When you open a channel to an application using `DDEInitiate`, you also specify a topic, such as a file name, to communicate with. The *item\$* is the part of the topic whose contents you are requesting.

`DDERequest` returns data as a text string. Data in any other format cannot be transferred, nor can graphics.

Example This example uses `DDEInitiate` to open a channel to Microsoft Word. It uses `DDERequest` to obtain a list of available topics (using the `System` topic). The example assumes that `WINWORD.EXE` is in the path `C:\MSOFFICE\WINWORD`.

```
Sub main
    Dim channel as Integer
    Dim appname as String
    Dim topic as String
    Dim item as String
    Dim msgtext as String
    Dim path as String
    appname="winword"
    topic="System"
    item="Topics"
    path="c:\msoffice\winword\"
    channel = -1
    x=Shell(path & appname & ".EXE")
    channel = DDEInitiate(appname, topic)
    If channel= -1 then
        msgtext="M/S Word not found -- please place on your path."
    Else
        On Error Resume Next
        msgtext="The Word topics available are:" & Chr$(13)
        msgtext=msgtext & Chr$(13) & DDERequest(channel,item)
        DDETerminate channel
        If Err<>0 then
            msgtext="DDE Access failed."
        End If
    End If
    MsgBox msgtext
End Sub
```

See Also

| | |
|-------------------------------|---------------------------|
| <code>DDEAppReturnCode</code> | <code>DDEPoke</code> |
| <code>DDEExecute</code> | <code>DDETerminate</code> |
| <code>DDEInitiate</code> | |

DDETerminate

Statement

Description Closes the specified dynamic data exchange (DDE) channel.

DDETerminate

Syntax **DDETerminate** *channel%*

| Syntax Element | Description |
|-----------------|---|
| <i>channel%</i> | An integer or expression for the open DDE channel number. |

Comments To free system resources, you should close channels you aren't using. If *channel%* does not correspond to an open channel, an error occurs.

Example This example uses DDEInitiate to open a channel to Microsoft Word. It uses DDERequest to obtain a list of available topics (using the System topic), and then terminates the channel using DDETerminate. The example assumes that WINWORD.EXE is in the path C:\MSOFFICE\WINWORD.

```
Sub main
  Dim channel as Integer
  Dim appname as String
  Dim topic as String
  Dim item as String
  Dim msgtext as String
  Dim path as String
  appname="winword"
  topic="System"
  item="Topics"
  path="c:\msoffice\winword\"
  channel = -1
  x=Shell(path & appname & ".EXE")
  channel = DDEInitiate(appname, topic)
  If channel= -1 then
    msgtext="M/S Word not found -- please place on your path."
  Else
    On Error Resume Next
    msgtext="The Word topics available are:" & Chr$(13)
    msgtext=msgtext & Chr$(13) & DDERequest(channel,item)
    DDETerminate channel
    If Err<>0 then
      msgtext="DDE Access failed."
    End If
  End If
  MsgBox msgtext
End Sub
```

See Also

| | |
|------------------|------------|
| DDEAppReturnCode | DDEPoke |
| DDEExecute | DDERequest |
| DDEInitiate | |

Declare Statement

Description Declares a procedure in a module or dynamic link library (DLL).

Syntax A **Declare Sub** *name* [*libSpecification*]
[(*arg* [As *type*],...)]

Syntax B **Declare Function** *name* [*libSpecification*]
[(*arg* [As *type*],...)] [As *functype*]

| Syntax Element | Description |
|-------------------------|---|
| <i>name</i> | The sub procedure or function procedure to declare. |
| <i>libSpecification</i> | The location of the procedure (module or DLL). |
| <i>arg</i> | An argument to pass to the procedure or function when it is called. Multiple arguments are separated by commas. |
| <i>type</i> | The data type of an argument in <i>arg</i> . |
| <i>functype</i> | The data type of the return value for a function procedure. |

Comments A Sub procedure does not return a value. A Function procedure returns a value and can be used in an expression. To specify the data type for the return value of a function, end the function name with a type declaration character or use the *As functype* clause shown above. If no type is provided, the return value defaults to data type Variant.

If the *libSpecification* is of the format:

```
BasicLib "libName" [Alias "aliasname"]
```

the procedure is in another SQABasic module (.sbl or .rec) named *libName*. The *Alias* keyword specifies that the procedure in *libName* is called *aliasname*. The other module will be loaded on demand whenever the procedure is called. SQABasic will not automatically unload modules that are loaded in this fashion. SQABasic will detect errors of mis-declaration.

If the *libSpecification* is of the format:

```
Lib "libName" [Alias ["ordinal["]] or  
Lib "libName" [Alias "aliasname"]
```

the procedure is in a Dynamic Link Library (DLL) named *libName*. The *ordinal* argument specifies the ordinal number of the procedure within the external DLL. Alternatively, *aliasname* specifies the name of the procedure within the external DLL. If neither *ordinal* nor *aliasname* is specified, the DLL function is accessed by name. It is recommended that the *ordinal* be used

Declare

whenever possible, since accessing functions by name might cause the module to load more slowly.

A forward declaration is needed only when a procedure in the current module is referenced before it is defined. In this case, the `BasicLib`, `Lib` and `Alias` clauses are not used.

arg contains an argument being passed to the sub procedure or function. An argument is represented by a variable name. Multiple arguments are separated by commas. Note the following information about the arguments being passed:

- ▶ The data type of an argument can be specified through a type declaration character or through the `As` clause.
- ▶ Arguments of a `User-Defined` data type are declared through an `As` clause and a *type* that has previously been defined through the `Type` statement.
- ▶ If an argument is an array, use empty parentheses after the argument name. The array dimensions are not specified within the `Declare` statement.

External DLL procedures are called with the PASCAL calling convention (the actual arguments are pushed on the stack from left to right). By default, the actual arguments are passed by `Far` reference. For external DLL procedures, there are two additional keywords, `ByVal` and `Any`, that can be used in the argument list.

When `ByVal` is used, it must be specified before the argument it modifies. When applied to numeric data types, `ByVal` indicates that the argument is passed by value, not by reference. When applied to string arguments, `ByVal` indicates that the string is passed by `Far` pointer to the string data. By default, strings are passed by `Far` pointer to a string descriptor.

`Any` can be used as a type specification, and permits a call to the procedure to pass a value of any data type. When `Any` is used, type checking on the actual argument used in calls to the procedure is disabled (although other arguments not declared as type `Any` are fully type-safe). The actual argument is passed by `Far` reference, unless `ByVal` is specified, in which case the actual value is placed on the stack (or a pointer to the string in the case of string data). `ByVal` can also be used in the call. It is the external DLL procedure's responsibility to determine the type and size of the passed-in value.

When an empty string ("") is passed `ByVal` to an external procedure, the external procedure will receive a valid (non-NULL) pointer to a character of 0. To send a NULL pointer, `Declare` the procedure argument as `ByVal As Any`, and call the procedure with an argument of 0.

Example This example declares a function that is later called by the main sub procedure. The function does nothing but set its return value to 1.

```

Declare Function SBL_exfunction()
Sub main
    Dim y as Integer
    Call SBL_exfunction
    y=SBL_exfunction
    MsgBox "The value returned by the function is: " & y
End Sub

Function SBL_exfunction()
    SBL_exfunction=1
End Function

```

See Also Call Dim
 Const Static
 Def`type` Type

Def`type` Statement

Description Declares the default data type for variables whose names start with the specified characters.

Syntax **DefCur** *letterrange* **DefInt** *letterrange*
 DefLng *letterrange* **DefSng** *letterrange*
 DefDbl *letterrange* **DefStr** *letterrange*
 DefVar *letterrange*

| Syntax Element | Description |
|--------------------|--|
| <i>letterrange</i> | The first letter of a variable name. The value can be a single letter, a comma-separated list of letters, or a range of letters. For example, a-d specifies a, b, c and d. |

Comments The case of the letters is not important, even in a letter range. The letter range a-z is treated as a special case. It denotes all alphabetic characters, including international characters.

The Def`type` statement affects only the module in which it is specified. It must precede any variable definition within the module.

DelayFor

The following table shows the variable type for each statement:

| Statement | Declares variables of type |
|-----------|----------------------------|
| DefCur | Currency |
| DefInt | Integer |
| DefLng | Long |
| DefSng | Single |
| DefDbl | Double |
| DefStr | String |
| DefVar | Variant |

Variables defined using the `Global` or `Dim` can override the `DefType` statement by using an `As` clause or a type character.

Example

This example finds the average of bowling scores entered by the user. Since the variable `average` begins with `a`, it is automatically defined as a single-precision floating point number. The other variables will be defined as Integers.

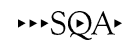
```
DefInt c,s,t
DefSng a
Sub main
  Dim count
  Dim total
  Dim score
  Dim average
  Dim msgtext
  For count=0 to 4
    score=InputBox("Enter bowling score #" & count+1 & ":")
    total=total+score
  Next count
  average=total/count
  msgtext="Your average is: " & average
  MsgBox msgtext
End Sub
```

See Also

Declare Let
Dim Type

DelayFor

Utility Command



Description Delays execution of the script for a specified number of milliseconds.

| Syntax | DelayFor <i>TimeInterval&</i> | | | | |
|--------------------------|--|----------------|-------------|--------------------------|--------------------------------|
| | <table border="1"> <thead> <tr> <th>Syntax Element</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i>TimeInterval&</i></td> <td>Time in milliseconds to delay.</td> </tr> </tbody> </table> | Syntax Element | Description | <i>TimeInterval&</i> | Time in milliseconds to delay. |
| Syntax Element | Description | | | | |
| <i>TimeInterval&</i> | Time in milliseconds to delay. | | | | |
| Comments | <p>This command pauses execution of the script for a specified period of time. During this time, Robot yields control to Windows, which may service other applications.</p> <p>This command corresponds to the Delay option in Robot's Wait States menu.</p> <p>For Wait parameters in verification points, see the individual verification point (VP) commands.</p> | | | | |
| Example | <p>This example pauses playback for 2000 milliseconds, or 2 seconds.</p> <pre>DelayFor 2000</pre> | | | | |
| See Also | None. | | | | |

Desktop

User Action Command

»»SQA»

Description Performs an action on the Windows desktop.

Syntax **Desktop** *action%*, *parameters\$*

| Syntax Element | Description |
|----------------|--|
| <i>action%</i> | <p>One of these mouse actions:</p> <ul style="list-style-type: none"> ▶ <i>MouseClicked</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> must contain <i>Coords=x,y</i>. ▶ <i>MouseDown</i>. The dragging of the mouse while mouse buttons and/or shifting keys (Ctrl, Alt, Shift) are pressed. When <i>action%</i> contains a mouse-drag value, <i>parameters\$</i> must contain <i>Coords=x1,y1,x2,y2</i>. <p>See Appendix E for a list of mouse click and drag values.</p> |

▶ ▶ ▶

Dialog (Function)

▶ ▶ ▶

| Syntax Element | Description |
|----------------------|---|
| <i>parameters</i> \$ | Valid values: <ul style="list-style-type: none">▶ <i>Coords=x,y</i>. If <i>action%</i> is a mouse click, specifies the coordinates of the click, relative to the top left of the object.▶ <i>Coords=x1,y1,x2,y2</i>. If <i>action%</i> is a mouse drag, specifies the coordinates, where <i>x1,y1</i> are the starting coordinates of the drag, and <i>x2,y2</i> are the ending coordinates. The coordinates are relative to the top left of the object. |

Comments No recognition methods are used to identify the desktop object type because there is only one Windows desktop and it is automatically recognized.

Since screen coordinates are used in this statement, it fails after the automatic timeout period if a window obscures the specified position on the desktop.

Example This example double-clicks the desktop at *x,y* coordinates of 306,223. (Double-clicking the desktop accesses the Windows Task List.)

```
Desktop DblClick, "Coords=306,223"
```

This example performs a left drag against the desktop at the designated *x1,y1,x2,y2* coordinates.

```
Desktop Left_Drag, "Coords=219,335,118,326"
```

See Also ComboBox EditBox
ComboBox ListBox

Dialog Function

Description Displays a dialog box and returns a number for the button selected (-1=OK, 0=Cancel).

Syntax **Dialog** (*recordName*)

| Syntax Element | Description |
|-------------------|--|
| <i>recordName</i> | A variable name declared as a dialog box record. |

Comments If the dialog box contains additional command buttons (for example, Help), the `Dialog` function returns a number greater than 0. 1 corresponds to the first command button, 2 to the second, and so on.

The dialog box *recordName* must have been declared using the `Dim` statement with the `As` parameter followed by a dialog box definition name. This name comes from the name argument used in the `Begin Dialog` statement.

To trap a user's selections within a dialog box, you must create a function and specify it as the last argument to the `Begin Dialog` statement. See `Begin Dialog` for more information.

The `Dialog` function does not return until the dialog box is closed.

Example This example creates a dialog box with a drop down combo box in it and three buttons: OK, Cancel, and Help. The `Dialog` function used here enables the sub procedure to trap when the user clicks on any of these buttons.

```
Sub main
  Dim cchoices as String
  Dim answer as Integer
  cchoices="All"+Chr$(9)+"Nothing"
  Begin Dialog UserDialog 180, 95, "SQABasic Dialog Box"
    ButtonGroup .ButtonGroup1
    Text 9, 3, 69, 13, "Filename:", .Text1
    ComboBox 9, 17, 111, 41, cchoices, .ComboBox1
    OKButton 131, 8, 42, 13
    CancelButton 131, 27, 42, 13
    PushButton 132, 48, 42, 13, "Help", .Push1
  End Dialog
  Dim mydialogbox As UserDialog
  answer= Dialog(mydialogbox)
  Select Case answer
    Case -1
      MsgBox "You pressed OK"
    Case 0
      MsgBox "You pressed Cancel"
    Case 1
      MsgBox "You pressed Help"
  End Select
End Sub
```

See Also `Begin Dialog` `Dialog` statement
`End Dialog`

Dialog Statement

Description Displays a dialog box.

Syntax **Dialog** *recordName*

| Syntax Element | Description |
|-------------------|--|
| <i>recordName</i> | A variable name declared as a dialog box record. |

Comments The dialog box *recordName* must have been declared using the `Dim` statement with the `As` parameter followed by a dialog box definition name. This name comes from the name argument used in the `Begin Dialog` statement.

If the user exits the dialog box by pushing the Cancel button, the runtime error 102 is triggered, which can be trapped using `On Error`.

To trap a user's selections within a dialog box, you must create a function and specify it as the last argument to the `Begin Dialog` statement. See `Begin Dialog` for more information.

The `Dialog` statement does not return until the dialog box is closed.

Example This example defines and displays a dialog box defined as *UserDialog* and named *mydialogbox*. If the user presses the Cancel button, an error code of 102 is returned and is trapped by the `If . . . Then` statement listed after the `Dialog` statement.

```
Sub main
  Dim cchoices as String
  On Error Resume Next
  cchoices="All"+Chr$(9)+"Nothing"
  Begin Dialog UserDialog 180, 95, "SQABasic Dialog Box"
    ButtonGroup .ButtonGroup1
    Text 9, 3, 69, 13, "Filename:", .Text1
    ComboBox 9, 17, 111, 41, cchoices, .ComboBox1
    OKButton 131, 8, 42, 13
    CancelButton 131, 27, 42, 13
  End Dialog
  Dim mydialogbox As UserDialog
  Dialog mydialogbox
  If Err=102 then
    MsgBox "You pressed Cancel."
  Else
    MsgBox "You pressed OK."
  End If
End Sub
```

See Also `Begin Dialog`
`End Dialog`
`Dialog` statement

Dim Statement

Description Declares variables for use in an SQABasic program.

Syntax **Dim** [Shared] *variableName* [As [New] *type*] [,*variableName* [As [New] *type*]] ...

| Syntax Element | Description | | | | | | | | | | |
|---------------------|--|---------|-------------------|------|--------------------------------|--------|--------|--------|---------|----------|--|
| <i>variableName</i> | The name of the variable to declare. | | | | | | | | | | |
| <i>type</i> | <p>The data type of the variable. Valid values include:</p> <table> <tbody> <tr> <td>Integer</td> <td>String (variable)</td> </tr> <tr> <td>Long</td> <td>String * <i>length</i> (fixed)</td> </tr> <tr> <td>Single</td> <td>Object</td> </tr> <tr> <td>Double</td> <td>Variant</td> </tr> <tr> <td>Currency</td> <td></td> </tr> </tbody> </table> <p>In addition, you can specify any User-Defined data type, including a dialog box record.</p> | Integer | String (variable) | Long | String * <i>length</i> (fixed) | Single | Object | Double | Variant | Currency | |
| Integer | String (variable) | | | | | | | | | | |
| Long | String * <i>length</i> (fixed) | | | | | | | | | | |
| Single | Object | | | | | | | | | | |
| Double | Variant | | | | | | | | | | |
| Currency | | | | | | | | | | | |

Comments *variableName* must begin with a letter and contain only letters, numbers and underscores. A name can also be delimited by brackets, and any character can be used inside the brackets, except for other brackets.

```
Dim my_1st_variable As String
Dim [one long and strange! variable name] As String
```

Basic is a strongly typed language. All variables must be assigned a data type or they will be automatically assigned the data type Variant.

If the *As* clause is not used, the *type* of the variable can be specified by using a type-declaration character as a suffix to *variableName*. The two different type-specification methods can be intermixed in a single *Dim* statement (although not on the same variable).

Regardless of which mechanism you use to declare a global variable, you can choose to use or omit the type-declaration character when referring to the variable in the rest of your program. The type suffix is not considered part of the variable name.

Dim

Arrays

Arrays support all SQABasic data types. Arrays of arrays and dialog box records are not supported.

Array variables are declared by including a subscript list as part of the *variableName*. The syntax to use for *variableName* is:

```
Dim variable([subscriptRange, ... ]) As typeName or  
Dim variable_with_suffix([subscriptRange, ... ])
```

where *subscriptRange* is of the format:

```
[startSubscript To] endSubscript
```

If *startSubscript* is not specified, 0 is used as the default. The `Option Base` statement can be used to change the default.

Both the *startSubscript* and the *endSubscript* are valid subscripts for the array. The maximum number of subscripts that can be specified in an array definition is 60. The maximum total size for an array is only limited by the amount of memory available.

If no *subscriptRange* is specified for an array, the array is declared as a dynamic array. In this case, the `ReDim` statement must be used to specify the dimensions of the array before the array can be used.

Numbers

Numeric variables can be declared using the `As` clause and one of the following numeric types: `Currency`, `Integer`, `Long`, `Single`, `Double`. Numeric variables can also be declared by including a type character as a suffix to the name. Numeric variables are initialized to 0.

Objects

Object variables are declared using an `As` clause and a *typeName* of a class. Object variables can be `Set` to refer to an object, and then used to access members and methods of the object using dot notation.

```
Dim OLE2 As Object  
Set OLE2 = CreateObject("spoly.cpoly")  
OLE2.reset
```

An object can be declared as `New` for some classes. In such instances, the object variable does not need to be `Set`; a new object will be allocated when the variable is used.

```
Dim variableName As New className  
variableName.methodName
```

Note: The class `Object` does not support the `New` operator.

Strings

SQABasic supports two types of strings: fixed-length and dynamic. Fixed-length strings are declared with a specific length (between 1 and 32,767) and cannot be changed later. Use the following syntax to declare a fixed-length string:

```
Dim variableName As String*length
```

Dynamic strings have no declared length, and can vary in length from 0 to 32,767. The initial length for a dynamic string is 0. Use the following syntax to declare a dynamic string:

```
Dim variableName$ or
Dim variableName As String
```

When initialized, fixed-length strings are filled with zeros. Dynamic strings are initialized as zero-length strings.

User-Defined

Variables of a user-defined type are declared by using an `AS` clause and a `typeName` that has been defined previously using the `TYPE` statement. The syntax is:

```
Dim variableName As typeName
```

Variables of a user-defined type are made up of a collection of data elements called fields. These fields can be of any numeric, string, Variant, or other user-defined type. See `TYPE` for details on accessing fields within a user-defined type.

You can also use the `Dim` statement to declare an instance of a dialog box record. In this case, `typeName` is specified as `dialogName`, where `dialogName` matches a dialog box record previously defined using `Begin Dialog`. The declared dialog box variable can then be used in a `Dialog` statement.

Variants

Declare variables as Variants when the type of the variable is not known at the start of, or might change during, the procedure. For example, a Variant is useful for holding input from a user when valid input can be either text or numbers. Use the following syntax to declare a Variant:

```
Dim variableName or
Dim variableName As Variant
```

Variant variables are initialized to `VarType Empty`.

Variables can be shared across modules. A variable declared inside a procedure has scope `Local` to that procedure. A variable declared outside a procedure has scope `Local` to the module. If you declare a variable with the same name as a module variable, the module variable is not accessible. See the `Global` statement for details.

The `Shared` keyword is included for backward compatibility with older versions of Basic. It is not allowed in `Dim` statements inside a procedure. It has no effect.

Dir

It is considered good programming practice to declare all variables. To force all variables to be explicitly declared use the `Option Explicit` statement. It is also recommended that you place all procedure-level `Dim` statements at the beginning of the procedure.

Regardless of which mechanism you use to declare a variable, you can choose to use or omit the type character when referring to the variable in the rest of your program. The type suffix is not considered part of the variable name.

Example This example shows a `Dim` statement for each of the possible data types.

```
Rem Must define a user-defined type before you can declare a
variable of that type
Type TestType
    Custno As Integer
    Custname As String
End Type

Sub main
    Dim counter As Integer
    Dim fixedstring As String*25
    Dim varstring As String
    Dim MyType As TestType
    Dim ole2var As Object
    Dim F(1 to 10), A()
    ...      Code here
End Sub
```

See Also

| | |
|-------------|--------|
| Global | Set |
| Option Base | Static |
| ReDim | Type |

Dir Function

Description Returns a file name that matches the specified pattern.

Syntax `Dir[$] [(pathname$ [,attributes%])]`

| Syntax Element | Description |
|-------------------|--|
| \$ | Optional. If specified the return type is <code>String</code> . If omitted the function will return a <code>Variant of VarType 8 (String)</code> . |
| <i>pathname\$</i> | A string expression identifying a path or file name. |

► ► ►

▶ ▶ ▶

| Syntax Element | Description |
|--------------------|---|
| <i>attributes%</i> | An integer expression specifying the file attributes to select. Valid attributes: <ol style="list-style-type: none"> 0. Return normal files 2. Add hidden files 4. Add system files 8. Return volume label 16. Add directories |

Comments *Pathname\$* can include a drive specification and wildcard characters (? and *). *Dir* returns the first file name that matches the *pathname\$* argument. An empty string ("") passed as *pathname\$* is interpreted as the current directory (same as "."). To retrieve additional matching file names, call the *Dir* function again, omitting the *pathname\$* and *attributes%* arguments. If no file is found, an empty string ("") is returned.

The default value for *attributes%* is 0. In this case, *Dir* returns only files without directory, hidden, system, or volume label attributes set.

The *attributes%* values can be added together to select multiple attributes. For example, to list hidden and system files in addition to normal files set *attributes%* to 6 (6=2+4).

If *attributes%* is set to 8, the *Dir* function returns the volume label of the drive specified in the *pathname\$*, or of the current drive if drive is not explicitly specified. If volume label attribute is set, all other attributes are ignored.

Example This example lists the contents of the diskette in drive A.

```
Sub main
  Dim msgret
  Dim directory, count
  Dim x, msgtext
  Dim A()
  msgret=MsgBox("Insert a disk in drive A.")
  count=1
  ReDim A(100)
  directory=Dir ("A:\*.*")
  Do While directory<>""
    A(count)=directory
    count=count+1
    directory=Dir
  Loop
  msgtext="Contents of drive A:\ is:" & Chr(10) & Chr(10)
  For x=1 to count
    msgtext=msgtext & A(x) & Chr(10)
  Next x
  MsgBox msgtext
End Sub
```

DlgControlID

See Also ChDir Mkdir
 ChDrive Rmdir
 CurDir

DlgControlID

Function

Description Returns the numeric ID of a dialog box control with the specified *Id\$* in the active dialog box.

Syntax **DlgControlID** (*Id\$*)

| Syntax Element | Description |
|----------------|-------------------------------------|
| <i>Id\$</i> | The string ID for a dialog control. |

Comments The **DlgControlID** function translates a string *Id\$* into a numeric ID. This function can only be used from within a dialog box function. The value of the numeric identifier is based on the position of the dialog box control with the dialog; it will be 0 (zero) for the first control, 1 (one) for the second control, and so on.

Given the following example, the statement `DlgControlID("doGo")` will return the value 1.

```
Begin Dialog newdlg 200, 200
  PushButton 40, 50, 80, 20, "&Stop", .doStop
  PushButton 40, 80, 80, 20, "&Go", .doGo
End Dialog
```

The advantage of using a dialog box control's numeric ID is that it is more efficient, and numeric values can sometimes be more easily manipulated.

Rearranging the order of a control within a dialog box will change its numeric ID. For example, if a `PushButton` control originally had a numeric value of 1, and a `TextBox` control is added before it, the `PushButton` control's new numeric value will be 2. This is shown in the following example:

```
CheckBox 40, 110, 80, 20, "CheckBox", .CheckBox1
TextBox 40, 20, 80, 20, .TextBox1      this is the new added control
PushButton 40, 80, 80, 20, "&Go", .doGo
```

The string IDs come from the last argument in the dialog definition statement that created the dialog control, such as the `TextBox` or `ComboBox` statements. The string ID does not include the period (.) and is case-sensitive.

Use **DlgControlID** only while a dialog box is running. See the `Begin Dialog` statement for more information.

Example This example displays a dialog box similar to File Open.

```

Declare Sub ListFiles(str1$)
Declare Function FileDlgFunction(identifier$, action, suppvalue)

Sub main
  Dim identifier$
  Dim action as Integer
  Dim suppvalue as Integer
  Dim filetypes as String
  Dim exestr$()
  Dim button as Integer
  Dim x as Integer
  Dim directory as String
  filetypes="Program files (*.exe)+Chr$(9)+ "All Files (*.*)"
  Begin Dialog newdlg 230, 145, "Open", .FileDlgFunction
    '$CStrings Save
    Text 8, 6, 60, 11, "&Filename:"
    TextBox 8, 17, 76, 13, .TextBox1
    ListBox 9, 36, 75, 61, exestr$(), .ListBox1
    Text 8, 108, 61, 9, "List Files of &Type:"
    DropListBox 7, 120, 78, 30, filetypes, .DropListBox1
    Text 98, 7, 43, 10, "&Directories:"
    Text 98, 20, 46, 8, "c:\windows"
    ListBox 99, 34, 66, 66, "", .ListBox2
    Text 98, 108, 44, 8, "Dri&ves:"
    DropListBox 98, 120, 68, 12, "", .DropListBox2
    OKButton 177, 6, 50, 14
    CancelButton 177, 24, 50, 14
    PushButton 177, 42, 50, 14, "&Help"
    '$CStrings Restore
  End Dialog
  Dim dlg As newdlg
  button = Dialog(dlg)
End Sub

Sub ListFiles(str1$)
  DlgText 1, str1$
  x=0
  Redim exestr$(x)
  directory=Dir$("c:\windows\" & str1$,16)
  If directory<>"" then
    Do
      exestr$(x)=LCase$(directory)
      x=x+1
      Redim Preserve exestr$(x)
      directory=Dir
    Loop Until directory=""
  End If
  DlgListBoxArray 2, exestr$()
End Sub

```

DlgEnable (Function)

```
Function FileDlgFunction(identifier$, action, suppvalue)
  Select Case action
    Case 1
      str1$="*.exe" 'dialog box initialized
      ListFiles str1$
    Case 2
      'button or control value changed
      If DlgControlID(identifier$) = 4 Then
        If DlgText(4)="All Files (*.*)" then
          str1$="*.*"
        Else
          str1$="*.exe"
        End If
      ListFiles str1$
      End If
    Case 3
      'text or combo box changed
      str1$=DlgText$(1)
      ListFiles str1$
    Case 4
      'control focus changed
    Case 5
      'idle
  End Select
End Function
```

See Also

| | |
|---------------------------|-------------------------|
| Begin Dialog | DlgSetPicture statement |
| End Dialog | DlgText function |
| DlgEnable function | DlgText statement |
| DlgEnable statement | DlgValue function |
| DlgFocus function | DlgValue statement |
| DlgFocus statement | DlgVisible function |
| DlgListBoxArray function | DlgVisible statement |
| DlgListBoxArray statement | |

DlgEnable

Function

Description Returns the enable state for the specified dialog control (1=enabled, 0=disabled).

Syntax **DlgEnable** (*Id*)

| Syntax Element | Description |
|----------------|--|
| <i>Id</i> | The control ID for the dialog control. |

Comments If a dialog box control is enabled, it is accessible to the user. You might want to disable a control if its use depends on the selection of other controls.

Use the `DlgControlID` function to find the numeric ID for a dialog control, based on its string identifier.

Use `DlgEnable` only while a dialog box is running. See the `Begin Dialog` statement for more information.

Example This example displays a dialog box with two check boxes, one labeled **Either**, the other labeled **Or**. If the user clicks on **Either**, the **Or** option is grayed. Likewise, if **Or** is selected, **Either** is grayed. The example uses the `DlgEnable` statement to toggle the state of the buttons.

```

Declare Function FileDlgFunction(identifier$, action, suppvalue)
Sub Main
    Dim button as integer
    Dim identifier$
    Dim action as Integer
    Dim suppvalue as Integer
    Begin Dialog newdlg 186,92,"DlgEnable example",.FileDlgFunction
        OKButton 130, 6, 50, 14
        CancelButton 130, 23, 50, 14
        CheckBox 34, 25, 75, 19, "Either", .CheckBox1
        CheckBox 34, 43, 73, 25, "Or", .CheckBox2
    End Dialog
    Dim dlg As newdlg
    button = Dialog(dlg)
End Sub

Function FileDlgFunction(identifier$, action, suppvalue)
    Select Case action
        Case 2 'button or control value changed
            If DlgControlID(identifier$) = 2 Then
                DlgEnable 3
            Else
                DlgEnable 2
            End If
        End Select
    End Function

```

See Also

| | |
|---------------------------|-------------------------|
| Begin Dialog | DlgSetPicture statement |
| End Dialog | DlgText function |
| DlgControlID function | DlgText statement |
| DlgEnable function | DlgValue function |
| DlgFocus function | DlgValue statement |
| DlgFocus statement | DlgVisible function |
| DlgListBoxArray function | DlgVisible statement |
| DlgListBoxArray statement | |

DlgEnable

Statement

Description Enables, disables, or toggles the state of the specified dialog control.

Syntax `DlgEnable Id [, mode]`

| Syntax Element | Description |
|----------------|---|
| <i>Id</i> | The control ID for the dialog control to change. |
| <i>mode</i> | An integer representing the enable state (1=enable, 0=disable). |

DlgEnable (Statement)

Comments If *mode* is omitted, the DlgEnable toggles the state of the dialog control specified by *Id*. If a dialog box control is enabled, it is accessible to the user. You might want to disable a control if its use depends on the selection of other controls.

Use the DlgControlID function to find the numeric ID for a dialog control, based on its string identifier. The string IDs come from the last argument in the dialog definition statement that created the dialog control, such as the TextBox or ComboBox statements.

Use DlgEnable only while a dialog box is running. See the Begin Dialog statement for more information.

Example This example displays a dialog box with one check box, labeled Show More, and a group box, labeled More, with two option buttons, Option 1 and Option 2. It uses the DlgEnable function to enable the More group box and its options if the Show More check box is selected.

```
Declare Function FileDlgFunction(identifier$, action, suppvalue)
Sub Main
    Dim button as integer
    Dim identifier$
    Dim action as Integer
    Dim suppvalue as Integer
    Begin Dialog newdlg 186,92,"DlgEnable example",.FileDlgFunction
        OKButton 130, 6, 50, 14
        CancelButton 130, 23, 50, 14
        CheckBox 13, 6, 75, 19, "Show more", .CheckBox1
        GroupBox 16, 28, 94, 50, "More"
        OptionGroup .OptionGroup1
            OptionButton 23, 40, 56, 12, "Option 1", .OptionButton1
            OptionButton 24, 58, 61, 13, "Option 2", .OptionButton2
    End Dialog
    Dim dlg As newdlg
    button = Dialog(dlg)
End Sub

Function FileDlgFunction(identifier$, action, suppvalue)
    Select Case action
        Case 1
            DlgEnable 3,0
            DlgEnable 4,0
            DlgEnable 5,0
        Case 2 'button or control value changed
            If DlgControlID(identifier$) = 2 Then
                If DlgEnable (3)=0 then
                    DlgEnable 3,1
                    DlgEnable 4,1
                    DlgEnable 5,1
                Else
                    DlgEnable 3,0
                    DlgEnable 4,0
                    DlgEnable 5,0
                End If
            End If
    End Select
End Function
```

| | | |
|----------|---------------------------|-------------------------|
| See Also | Begin Dialog | DlgSetPicture statement |
| | End Dialog | DlgText function |
| | DlgControlID function | DlgText statement |
| | DlgEnable statement | DlgValue function |
| | DlgFocus function | DlgValue statement |
| | DlgFocus statement | DlgVisible function |
| | DlgListBoxArray function | DlgVisible statement |
| | DlgListBoxArray statement | |

DlgEnd

Statement

Description Closes the active dialog box.

Syntax **DlgEnd** *exitCode*

| Syntax Element | Description |
|-----------------|--|
| <i>exitCode</i> | The return value after closing the dialog box (-1=OK, 0=Cancel). |

Comments *ExitCode* contains a return value only if the dialog box was displayed using the `Dialog` function. That is, if you used the `Dialog` statement, *exitCode* is ignored.

If the dialog box contains additional command buttons (for example, Help), the `Dialog` function returns a number greater than 0. 1 corresponds to the first command button, 2 to the second, and so on.

Use `DlgEnd` only while a dialog box is running. See the `Begin Dialog` statement for more information.

Example This example displays a dialog box with the message `You have 30 seconds to cancel`. The dialog box counts down from 30 seconds to 0. If the user clicks OK or Cancel during the countdown, the dialog box closes. If the countdown reaches 0, however, the `DlgEnd` statement closes the dialog box.

```
Function timeout(id$,action%,suppvalue&)
    Static timeoutStart as Long
    Static currentSecs as Long
    Dim thisSecs as Long
```

DlgEnd

```
Select Case action%
  Case 1
    'initialize the dialog box. Set the ticker value to 30
    'and remember when we put up the dialog box
    DlgText "ticker", "30"
    timeoutStart = timer
    currentSecs = 30
  Case 5
    'this is an idle message - set thisSecs to the number
    'of seconds left until timeout
    thisSecs = timer
    If thisSecs < timeoutStart Then thisSecs =
      thisSecs + 24*60*60
    thisSecs = 30 - (thisSecs - timeoutStart)
    ' if there are negative seconds left, timeout!
    If thisSecs < 0 Then DlgEnd -1
    ' If the seconds left has changed since last time,
    ' update the dialog box
    If thisSecs <> currentSecs Then
      DlgText "ticker", trim$(str$(thisSecs))
      currentSecs = thisSecs
    End If
    ' make sure to return non-zero so we keep getting
    ' idle messages
    timeout = 1
  End Select
End Function

Sub main
  Begin Dialog newdlg 167, 78, "Do You Want to Continue?", .timeout
    '$CStrings Save
    OKButton 27, 49, 50, 14
    CancelButton 91, 49, 50, 14
    Text 24, 14, 119, 8, "This is your last chance to bail out."
    Text 27, 30, 35, 8, "You have"
    Text 62, 30, 13, 8, "30", .ticker
    Text 74, 30, 66, 8, "seconds to cancel."
    '$CStrings Restore
  End Dialog
  Dim dlgVar As newdlg
  If dialog(dlgvar) = 0 Then
    Exit Sub ' abort
  End If
  ' do whatever it is we want to do
End Sub
```

See Also

| | |
|--------------------------|---------------------------|
| BeginDialog | DlgListBoxArray statement |
| End Dialog | DlgSetPicture statement |
| DlgControlID function | DlgText function |
| DlgEnable function | DlgText statement |
| DlgEnable statement | DlgValue function |
| DlgFocus function | DlgValue statement |
| DlgFocus statement | DlgVisible function |
| DlgListBoxArray function | DlgVisible statement |

DlgFocus

Function

Description Returns the control ID of the dialog control having the input focus.

Syntax **DlgFocus**[\$]()

Comments A control has focus when it is active and responds to keyboard input.
Use DlgFocus only while a dialog box is running. See the Begin Dialog statement for more information.

Example This example displays a dialog box with a check box, labeled Check1, and a text box, labeled Text Box 1, in it. When the box is initialized, the focus is set to the text box. As soon as the user clicks the check box, the focus goes to the OK button.

```

Declare Function FileDlgFunction(identifier$, action, suppvalue)
Sub main
  Dim button as integer
  Dim identifier$
  Dim action as Integer
  Dim suppvalue as Integer
  Begin Dialog newdlg 186, 92, "DlgFocus Example", .FileDlgFunction
    OKButton 130, 6, 50, 14
    CancelButton 130, 23, 50, 14
    TextBox 15, 37, 82, 12, .TextBox1
    Text 15, 23, 57, 10, "Text Box 1"
    CheckBox 15, 6, 75, 11, "Check1", .CheckBox1
  End Dialog
  Dim dlg As newdlg
  button = Dialog(dlg)
End Sub

Function FileDlgFunction(identifier$, action, suppvalue)
  Select Case action
    Case 1
      DlgFocus 2
    Case 2 'user changed control or clicked a button
      If DlgFocus() <> "OKButton" then
        DlgFocus 0
      End If
    End Select
  End Function

```

See Also

| | |
|---------------------------|-------------------------|
| Begin Dialog | DlgSetPicture statement |
| End Dialog | DlgText function |
| DlgControlID function | DlgText statement |
| DlgEnable function | DlgValue function |
| DlgEnable statement | DlgValue statement |
| DlgFocus statement | DlgVisible function |
| DlgListBoxArray function | DlgVisible statement |
| DlgListBoxArray statement | |

DlgFocus Statement

Description Sets the focus for the specified dialog control.

Syntax **DlgFocus** *Id*

| Syntax Element | Description |
|----------------|---|
| <i>Id</i> | The control ID for the dialog control to make active. |

Comments Use the `DlgControlID` function to find the numeric ID for a dialog control, based on its string identifier. The string IDs come from the last argument in the dialog definition statement that created the dialog control, such as the `TextBox` or `ComboBox` statements.

Use `DlgFocus` only while a dialog box is running. See the `Begin Dialog` statement for more information.

Example This example displays a dialog box with a check box, labeled `Check1`, and a text box, labeled `Text Box 1`, in it. When the box is initialized, the focus is set to the text box. As soon as the user clicks the check box, the focus goes to the OK button.

```

Declare Function FileDlgFunction(identifier$, action, supvalue)
Sub Main
    Dim button as integer
    Dim identifier$
    Dim action as Integer
    Dim supvalue as Integer
    Begin Dialog newdlg 186, 92, "DlgFocus Example", .FileDlgFunction
        OKButton 130, 6, 50, 14
        CancelButton 130, 23, 50, 14
        TextBox 15, 37, 82, 12, .TextBox1
        Text 15, 23, 57, 10, "Text Box 1"
        CheckBox 15, 6, 75, 11, "Check1", .CheckBox1
    End Dialog
    Dim dlg As newdlg
    button = Dialog(dlg)
End Sub

Function FileDlgFunction(identifier$, action, supvalue)
    Select Case action
        Case 1
            DlgFocus 2
        Case 2 'user changed control or clicked a button
            If DlgFocus() <> "OKButton" then
                DlgFocus 0
            End If
        End Select
    End Function

```

| | | |
|----------|---------------------------|-------------------------|
| See Also | BeginDialog | DlgSetPicture statement |
| | End Dialog | DlgText function |
| | DlgControlID function | DlgText statement |
| | DlgEnable function | DlgValue function |
| | DlgEnable statement | DlgValue statement |
| | DlgFocus function | DlgVisible function |
| | DlgListBoxArray function | DlgVisible statement |
| | DlgListBoxArray statement | |

DlgListBoxArray

Function

Description Returns the number of elements in a list or combo box.

Syntax **DlgListBoxArray** (*Id*[, *Array\$*])

| Syntax Element | Description |
|----------------|--|
| <i>Id</i> | The control ID for the list or combo box. |
| <i>Array\$</i> | The entries in the list box or combo box returned. |

Comments *Array\$* is a one-dimensional array of dynamic strings. If *array\$* is dynamic, its size is changed to match the number of strings in the list or combo box. If *array\$* is not dynamic and it is too small, an error occurs. If *array\$* is omitted, the function returns the number of entries in the specified dialog control.

Use the `DlgControlID` function to find the numeric ID for a dialog control, based on its string identifier. The string IDs come from the last argument in the dialog definition statement that created the dialog control, such as the `TextBox` or `ComboBox` statements.

Use `DlgListBoxArray` only while a dialog box is running. See the `BeginDialog` statement for more information.

DlgListBoxArray (Function)

Example This example displays a dialog box with a check box, labeled `Display List`, and an empty list box. If the user clicks the check box, the list box is filled with the contents of the array called `myarray`. The `DlgListBoxArray` function makes sure the list box is empty.

```
Declare Function FileDlgFunction(identifier$, action, suppvalue)
Sub Main
    Dim button as integer
    Dim identifier$
    Dim action as Integer
    Dim suppvalue as Integer
    Begin Dialog newdlg 186,92,"DlgListBoxArray Example",.FileDlgFunction
        'CStrings Save
        OKButton 130, 6, 50, 14
        CancelButton 130, 23, 50, 14
        ListBox 19, 26, 74, 59, "", .ListBox1
        CheckBox 12, 4, 86, 13, "Display List", .CheckBox1
        'CStrings Restore
    End Dialog
    Dim dlg As newdlg
    button = Dialog(dlg)
End Sub

Function FileDlgFunction(identifier$, action, suppvalue)
    Dim myarray$(3)
    Dim msgtext as Variant
    Dim x as Integer
    For x= 0 to 2
        myarray$(x)=Chr$(x+65)
    Next x
    Select Case action
        Case 1
        Case 2 'user changed control or clicked a button
            If DlgControlID(identifier$)=3 then
                If DlgListBoxArray(2)=0 then
                    DlgListBoxArray 2, myarray$()
                End If
            End If
        End Select
    End Function
```

See Also

| | |
|--|--------------------------------------|
| <code>BeginDialog</code> | <code>DlgSetPicture</code> statement |
| <code>End Dialog</code> | <code>DlgText</code> function |
| <code>DlgControlID</code> function | <code>DlgText</code> statement |
| <code>DlgEnable</code> function | <code>DlgValue</code> function |
| <code>DlgEnable</code> statement | <code>DlgValue</code> statement |
| <code>DlgFocus</code> function | <code>DlgVisible</code> function |
| <code>DlgFocus</code> statement | <code>DlgVisible</code> statement |
| <code>DlgListBoxArray</code> statement | |

DlgListBoxArray

Statement

Description Fills a list or combo box with an array of strings.

Syntax **DlgListBoxArray** *Id*, *Array\$*

| Syntax Element | Description |
|----------------|--|
| <i>Id</i> | The control ID for the list or combo box. |
| <i>Array\$</i> | The entries for the list box or combo box. |

Comments *Array\$* has to be a one-dimensional array of dynamic strings. One entry appears in the list box for each element of the array. If the number of strings changes depending on other selections made in the dialog box, you should use a dynamic array and `ReDim` the size of the array whenever it changes.

Use `DlgListBoxArray` only while a dialog box is running. See the `Begin Dialog` statement for more information.

Example This example displays a dialog box similar to File Open.

```

Declare Sub ListFiles(str1$)
Declare Function FileDlgFunction(identifrier$, action, suppvalue)

Sub main
  Dim identifrier$
  Dim action as Integer
  Dim suppvalue as Integer
  Dim filetypes as String
  Dim exestr$()
  Dim button as Integer
  Dim x as Integer
  Dim directory as String
  filetypes="Program files (*.exe)+Chr$(9)+"All Files (*.*)"
  Begin Dialog newdlg 230, 145, "Open", .FileDlgFunction
    '$CStrings Save
    Text 8, 6, 60, 11, "&Filename:"
    TextBox 8, 17, 76, 13, .TextBox1
    ListBox 9, 36, 75, 61, exestr$(), .ListBox1
    Text 8, 108, 61, 9, "List Files of &Type:"
    DropListBox 7, 120, 78, 30, filetypes, .DropListBox1
    Text 98, 7, 43, 10, "&Directories:"
    Text 98, 20, 46, 8, "c:\\windows"
    ListBox 99, 34, 66, 66, "", .ListBox2
    Text 98, 108, 44, 8, "Dri&ves:"
    DropListBox 98, 120, 68, 12, "", .DropListBox2
    OKButton 177, 6, 50, 14
    CancelButton 177, 24, 50, 14
    PushButton 177, 42, 50, 14, "&Help"
    '$CStrings Restore
  End Dialog
  Dim dlg As newdlg
  button = Dialog(dlg)
End Sub

```

DlgSetPicture

```
Sub ListFiles(str1$)
  DlgText 1,str1$
  x=0
  Redim exestr$(x)
  directory=Dir$("c:\windows\" & str1$,16)
  If directory<>" " then
    Do
      exestr$(x)=LCase$(directory)
      x=x+1
      Redim Preserve exestr$(x)
      directory=Dir
    Loop Until directory=""
  End If
  DlgListBoxArray 2,exestr$()
End Sub

Function FileDlgFunction(identifier$, action, suppvalue)
  Select Case action
    Case 1
      str1$="*.exe" 'dialog box initialized
      ListFiles str1$
    Case 2
      'button or control value changed
      If DlgControlId(identifier$) = 4 Then
        If DlgText(4)="All Files (*.*)" then
          str1$="*.*"
        Else
          str1$="*.exe"
        End If
      End If
      ListFiles str1$
    Case 3
      'text or combo box changed
      str1$=DlgText$(1)
      ListFiles str1$
    Case 4
      'control focus changed

    Case 5
      'idle
  End Select
End Function
```

| | | |
|----------|--------------------------|-------------------------|
| See Also | BeginDialog | DlgSetPicture statement |
| | End Dialog | DlgText function |
| | DlgControlID function | DlgText statement |
| | DlgEnable function | DlgValue function |
| | DlgFocus function | DlgValue statement |
| | DlgFocus statement | DlgVisible function |
| | DlgListBoxArray function | DlgVisible statement |
| | DlgEnable | |

DlgSetPicture

Statement

Description Changes the picture in a picture dialog control for the current dialog box.

Syntax **DlgSetPicture** *Id*, *filename\$*, *type*

| Syntax Element | Description |
|-------------------|--|
| <i>Id</i> | The control ID for the picture dialog control. |
| <i>filename\$</i> | The name of the bitmap file (.BMP) to use. |
| <i>type</i> | An integer representing the location of the file (0= <i>filename\$</i> , 3=Clipboard) |

Comments Use the `DlgControlID` function to find the numeric ID for a dialog control, based on its string identifier. The string IDs come from the last argument in the dialog definition statement that created the dialog control, such as the `TextBox` or `ComboBox` statements.

Use `DlgListBoxArray` only while a dialog box is running. See the `Begin Dialog` statement for more information.

See the `Picture` statement for more information about displaying pictures in dialog boxes.

Example This example displays a picture in a dialog box and changes the picture if the user clicks the check box labeled Change Picture. The example assumes the picture bitmaps are in the `C:\WINDOWS` directory.

```

Declare Function FileDlgFunction(identifier$, action, suppvalue)
Sub Main
    Dim button as integer
    Dim identifier$
    Dim action as Integer
    Dim suppvalue as Integer
    Begin Dialog newdlg 186,92,"DlgSetPicture Example",.FileDlgFunction
        OKButton 130, 6, 50, 14
        CancelButton 130, 23, 50, 14
        Picture 43, 28, 49, 31, "C:\WINDOWS\CIRCLES.BMP", 0
        CheckBox 30, 8, 62, 15, "Change Picture", .CheckBox1
    End Dialog
    Dim dlg As newdlg
    button = Dialog(dlg)
End Sub

Function FileDlgFunction(identifier$, action, suppvalue)
    Select Case action
        Case 1
        Case 2 'user changed control or clicked a button
            If DlgControlID(identifier$)=3 then
                If suppvalue=1 then
                    DlgSetPicture 2, "C:\WINDOWS\TILES.BMP",0
                Else
                    DlgSetPicture 2, "C:\WINDOWS\CIRCLES.BMP",0
                End If
            End If
        End Select
    End Function

```

DlgText (Function)

| | | |
|----------|---------------------------|----------------------|
| See Also | BeginDialog | DlgFocus statement |
| | End Dialog | DlgText function |
| | DlgControlID function | DlgText statement |
| | DlgEnable function | DlgValue function |
| | DlgEnable statement | DlgValue statement |
| | DlgListBoxArray function | DlgVisible function |
| | DlgListBoxArray statement | DlgVisible statement |
| | DlgFocus function | |
| | | |

DlgText Function

Description Returns the text associated with a dialog control for the current dialog box.

Syntax **DlgText**[\$] (*Id*)

| Syntax Element | Description |
|----------------|--------------------------------------|
| <i>Id</i> | The control ID for a dialog control. |

Comments If the control is a text box or a combo box, `DlgText` function returns the text that appears in the text box. If it is a list box, the function returns its current selection. If it is a text box, `DlgText` returns the text. If the control is a command button, option button, option group, or a check box, the function returns its label.

Use `DlgText` only while a dialog box is running. See the `Begin Dialog` statement for more information.

Example This example displays a dialog box similar to File Open. It uses `DlgText` to determine what group of files to display.

```
Declare Sub ListFiles(str1$)
Declare Function FileDlgFunction(identifier$, action, suppvalue)

Sub main
  Dim identifier$
  Dim action as Integer
  Dim suppvalue as Integer
  Dim filetypes as String
  Dim exestr$()
  Dim button as Integer
  Dim x as Integer
  Dim directory as String
  filetypes="Program files (*.exe)+Chr$(9)+"All Files (*.*)"
End Sub
```


DlgText (Function)

```
Begin Dialog newdlg 230, 145, "Open", .FileDlgFunction
  '$CStrings Save
  Text 8, 6, 60, 11, "&Filename:"
  TextBox 8, 17, 76, 13, .TextBox1
  ListBox 9, 36, 75, 61, exestr$, .ListBox1
  Text 8, 108, 61, 9, "List Files of &Type:"
  DropListBox 7, 120, 78, 30, filetype$, .DropListBox1
  Text 98, 7, 43, 10, "&Directories:"
  Text 98, 20, 46, 8, "c:\\windows"
  ListBox 99, 34, 66, 66, "", .ListBox2
  Text 98, 108, 44, 8, "Dri&ves:"
  DropListBox 98, 120, 68, 12, "", .DropListBox2
  OKButton 177, 6, 50, 14
  CancelButton 177, 24, 50, 14
  PushButton 177, 42, 50, 14, "&Help"
  '$CStrings Restore
End Dialog
Dim dlg As newdlg
button = Dialog(dlg)
End Sub

Sub ListFiles(str1$)
  DlgText 1, str1$
  x=0
  Redim exestr$(x)
  directory=Dir$("c:\\windows\\" & str1$,16)
  If directory<>"" then
    Do
      exestr$(x)=LCase$(directory)
      x=x+1
      Redim Preserve exestr$(x)
      directory=Dir
    Loop Until directory=""
  End If
  DlgListBoxArray 2, exestr$()
End Sub

Function FileDlgFunction(identifier$, action, supvalue)
  Select Case action
    Case 1
      str1$="*.exe" 'dialog box initialized
      ListFiles str1$
    Case 2 'button or control value changed
      If DlgControlId(identifier$) = 4 Then
        If DlgText(4)="All Files (*.*)" then
          str1$="*.*"
        Else
          str1$="*.exe"
        End If
      ListFiles str1$
      End If
    Case 3 'text or combo box changed
      str1$=DlgText$(1)
      ListFiles str1$
    Case 4 'control focus changed

    Case 5 'idle
  End Select
End Function
```

DlgText (Statement)

| | | | |
|----------|---------------------------|-------------------------|--|
| See Also | BeginDialog | DlgFocus function | |
| | End Dialog | DlgSetPicture statement | |
| | DlgControlID function | DlgText statement | |
| | DlgEnable function | DlgValue function | |
| | DlgEnable statement | DlgValue statement | |
| | DlgListBoxArray function | DlgVisible function | |
| | DlgListBoxArray statement | DlgVisible statement | |
| | DlgFocus statement | | |
| | | | |
| | | | |

DlgText Statement

Description Changes the text associated with a dialog control for the current dialog box.

Syntax **DlgText** *Id*, *text\$*

| Syntax Element | Description |
|----------------|---|
| <i>Id</i> | The control ID for a dialog control. |
| <i>text\$</i> | The text to use for the dialog control. |

Comments If the dialog control is a text box or a combo box, DlgText sets the text that appears in the text box. If it is a list box, a string equal to *text\$* or beginning with *text\$* is selected. If the dialog control is a text control, DlgText sets it to *text\$*. If the dialog control is a command button, option button, option group, or a check box, the statement sets its label.

The DlgText statement does not change the identifier associated with the control.

Use DlgText only while a dialog box is running. See the BeginDialog statement for more information.

Example This example displays a dialog box similar to File Open. It uses the DlgText statement to display the list of files in the Filename list box.

```
Declare Sub ListFiles(str1$)
Declare Function FileDlgFunction(identifier$, action, suppvalue)

Sub main
  Dim identifier$
  Dim action as Integer
  Dim suppvalue as Integer
  Dim filetypes as String
  Dim exestr$()
  Dim button as Integer
  Dim x as Integer
  Dim directory as String
  filetypes="Program files (*.exe)+Chr$(9)+"All Files (*.*)"

```

DlgText (Statement)

```

Begin Dialog newdlg 230, 145, "Open", .FileDlgFunction
  '$CStrings Save
  Text 8, 6, 60, 11, "&Filename:"
  TextBox 8, 17, 76, 13, .TextBox1
  ListBox 9, 36, 75, 61, exestr$, .ListBox1
  Text 8, 108, 61, 9, "List Files of &Type:"
  DropListBox 7, 120, 78, 30, filetype$, .DropListBox1
  Text 98, 7, 43, 10, "&Directories:"
  Text 98, 20, 46, 8, "c:\\windows"
  ListBox 99, 34, 66, 66, "", .ListBox2
  Text 98, 108, 44, 8, "Dri&ves:"
  DropListBox 98, 120, 68, 12, "", .DropListBox2
  OKButton 177, 6, 50, 14
  CancelButton 177, 24, 50, 14
  PushButton 177, 42, 50, 14, "&Help"
  '$CStrings Restore
End Dialog
Dim dlg As newdlg
button = Dialog(dlg)
End Sub

Sub ListFiles(str1$)
  DlgText 1, str1$
  x=0
  Redim exestr$(x)
  directory=Dir$("c:\\windows\\" & str1$,16)
  If directory<>"" then
    Do
      exestr$(x)=LCase$(directory)
      x=x+1
      Redim Preserve exestr$(x)
      directory=Dir
    Loop Until directory=""
  End If
  DlgListBoxArray 2, exestr$()
End Sub

Function FileDlgFunction(identifier$, action, supvalue)
  Select Case action
    Case 1
      str1$="*.exe" 'dialog box initialized
      ListFiles str1$
    Case 2 'button or control value changed
      If DlgControlId(identifier$) = 4 Then
        If DlgText(4)="All Files (*.*)" then
          str1$="*.*"
        Else
          str1$="*.exe"
        End If
      ListFiles str1$
      End If
    Case 3 'text or combo box changed
      str1$=DlgText$(1)
      ListFiles str1$
    Case 4 'control focus changed

    Case 5 'idle
  End Select
End Function

```

DlgValue (Function)

See Also

| | |
|---------------------------|-------------------------|
| BeginDialog | DlgFocus statement |
| End Dialog | DlgSetPicture statement |
| DlgControlID function | DlgText function |
| DlgEnable function | DlgValue function |
| DlgEnable statement | DlgValue statement |
| DlgListBoxArray function | DlgVisible function |
| DlgListBoxArray statement | DlgVisible statement |
| DlgFocus function | |

DlgValue

Function

Description Returns a numeric value for the state of a dialog control for the current dialog box.

Syntax **DlgValue** (*Id*)

| Syntax Element | Description |
|----------------|--|
| <i>Id</i> | <p>The control ID for a dialog control.</p> <p>The values returned depend on the type of dialog control:</p> <ul style="list-style-type: none">▶ CheckBox 1 = Selected, 0=Cleared, -1=Grayed▶ Option Group 0 = 1st button selected, 1 = 2nd button selected, etc.▶ ListBox 0 = 1st item, 1= 2nd item, etc.▶ ComboBox 0 = 1st item, 1 = 2nd item, etc.▶ Text, Textbox, Button Error occurs |

Comments Use DlgValue only while a dialog box is running. See the Begin Dialog statement for more information.

Example

This example changes the picture in the dialog box if the check box is selected and changes the picture to its original bitmap if the check box is turned off. The example assumes the picture bitmaps are in the C:\WINDOWS directory.

```

Declare Function FileDlgFunction(identifiier$, action, suppvalue)
Sub Main
  Dim button as integer
  Dim identifiier$
  Dim action as Integer
  Dim suppvalue as Integer
  Begin Dialog newdlg 186,92,"DlgSetPicture Example",.FileDlgFunction
    OKButton 130, 6, 50, 14
    CancelButton 130, 23, 50, 14
    Picture 43, 28, 49, 31, "C:\WINDOWS\CIRCLES.BMP", 0
    CheckBox 30, 8, 62, 15, "Change Picture", .CheckBox1
  End Dialog
  Dim dlg As newdlg
  button = Dialog(dlg)
End Sub

Function FileDlgFunction(identifiier$, action, suppvalue)
  Select Case action
  Case 1
  Case 2 'user changed control or clicked a button
    If DlgControlID(identifiier$)=3 then
      If DlgValue(3)=1 then
        DlgSetPicture 2, "C:\WINDOWS\TILES.BMP",0
      Else
        DlgSetPicture 2, "C:\WINDOWS\CIRCLES.BMP",0
      End If
    End If
  End Select
End Function

```

See Also

| | |
|---------------------------|-------------------------|
| BeginDialog | DlgFocus statement |
| End Dialog | DlgSetPicture statement |
| DlgControlID function | DlgText function |
| DlgEnable function | DlgText statement |
| DlgEnable statement | DlgValue statement |
| DlgListBoxArray function | DlgVisible function |
| DlgListBoxArray statement | DlgVisible statement |
| DlgFocus function | |

DlgValue (Statement)

DlgValue Statement

Description Changes the value associated with the dialog control for the current dialog box.

Syntax **DlgValue** *Id*, *value%*

| Syntax Element | Description |
|----------------|---|
| <i>Id</i> | The control ID for a dialog control. |
| <i>value%</i> | The new value for the dialog control. The values you use to set the control depend on the type of the control: <ul style="list-style-type: none">▶ CheckBox 1 = Select, 0=Clear, -1=Gray.▶ Option Group 0 = Select 1st button, 1 = Select 2nd button.▶ ListBox 0 = Select 1st item, 1= Select 2nd item, etc.▶ ComboBox 0 = Select 1st item, 1 = Select 2nd item, etc.▶ Text, Textbox, Button Error occurs |

Comments Use DlgValue only while a dialog box is running. See the Begin Dialog statement for more information.

Example This example displays a dialog box with a check box, labeled Change Option, and a group box with two option buttons, labeled Option 1 and Option 2. When the user clicks the Change Option button, Option 2 is selected.

```
Declare Function FileDlgFunction(identifier$, action, suppvalue)
Sub Main
  Dim button as integer
  Dim identifier$
  Dim action as Integer
  Dim suppvalue as Integer
  Begin Dialog newdlg 186, 92, "DlgValue Example", .FileDlgFunction
    OKButton 130, 6, 50, 14
    CancelButton 130, 23, 50, 14
    CheckBox 30, 8, 62, 15, "Change Option", .CheckBox1
    GroupBox 28, 34, 79, 47, "Group"
    OptionGroup .OptionGroup1
      OptionButton 41, 47, 52, 10, "Option 1", .OptionButton1
      OptionButton 41, 62, 58, 11, "Option 2", .OptionButton2
  End Dialog
  Dim dlg As newdlg
  button = Dialog(dlg)
End Sub
```

```

Function FileDlgFunction(identifier$, action, suppvalue)
  Select Case action
    Case 1
    Case 2 'user changed control or clicked a button
      If DlgControlID(identifier$)=2 then
        If DlgValue(2)=1 then
          DlgValue 4,1
        Else
          DlgValue 4,0
        End If
      End If
    End Select
  End Function

```

See Also

| | |
|---------------------------|-------------------------|
| BeginDialog | DlgFocus statement |
| End Dialog | DlgSetPicture statement |
| DlgControlID function | DlgText function |
| DlgEnable function | DlgText statement |
| DlgEnable statement | DlgValue function |
| DlgListBoxArray function | DlgVisible function |
| DlgListBoxArray statement | DlgVisible statement |
| DlgFocus function | |

DlgVisible

Function

Description Returns -1 if a dialog control is visible, 0 if it is hidden.

Syntax **DlgVisible** (*Id*)

| Syntax Element | Description |
|----------------|--------------------------------------|
| <i>Id</i> | The control ID for a dialog control. |

Comments Use DlgVisible only while a dialog box is running. See the BeginDialog statement for more information.

Example This example displays Option 2 in the Group box if the user clicks the check box labeled Show Option 2. If the user clicks the box again, Option 2 is hidden.

```

Declare Function FileDlgFunction(identifier$, action, suppvalue)
Sub Main
  Dim button as integer
  Dim identifier$
  Dim action as Integer
  Dim suppvalue as Integer

```

DlgVisible (Statement)

```
Begin Dialog newdlg 186,92,"DlgVisible Example",.FileDlgFunction
  OKButton 130, 6, 50, 14
  CancelButton 130, 23, 50, 14
  CheckBox 30, 8, 62, 15, "Show Option 2", .CheckBox1
  GroupBox 28, 34, 79, 47, "Group"
  OptionGroup .OptionGroup1
    OptionButton 41, 47, 52, 10, "Option 1", .OptionButton1
    OptionButton 41, 62, 58, 11, "Option 2", .OptionButton2
End Dialog
Dim dlg As newdlg
button = Dialog(dlg)
End Sub

Function FileDlgFunction(identifier$, action, supvalue)
  Select Case action
    Case 1
      DlgVisible 6,0
    Case 2 'user changed control or clicked a button
      If DlgControlID(identifier$)=2 then
        If DlgVisible(6)<>1 then
          DlgVisible 6
        End If
      End If
    End Select
  End Function
```

See Also

| | |
|---------------------------|-------------------------|
| BeginDialog | DlgFocus statement |
| End Dialog | DlgSetPicture statement |
| DlgControlID function | DlgText function |
| DlgEnable function | DlgText statement |
| DlgEnable statement | DlgValue function |
| DlgListBoxArray function | DlgValue statement |
| DlgListBoxArray statement | DlgVisible statement |
| DlgFocus function | |

DlgVisible Statement

Description Hides or displays a dialog control for the current dialog box.

Syntax **DlgVisible** *Id* [, *mode*]

| Syntax Element | Description |
|----------------|--|
| <i>Id</i> | The control ID for a dialog control. |
| <i>mode</i> | Value to use to set the dialog control state: 1. Display a previously hidden control. 0. Hide the control. |

Comments If you omit the *mode*, the dialog box state is toggled between visible and hidden. Use `DlgVisible` only while a dialog box is running. See the `Begin Dialog` statement for more information.

Example This example displays Option 2 in the Group box if the user clicks the check box labeled Show Option 2. If the user clicks the box again, Option 2 is hidden.

```

Declare Function FileDlgFunction(identifier$, action, suppvale)
Sub Main
  Dim button as integer
  Dim identifier$
  Dim action as Integer
  Dim suppvale as Integer
  Begin Dialog newdlg 186,92,"DlgVisible Example",.FileDlgFunction
    OKButton 130, 6, 50, 14
    CancelButton 130, 23, 50, 14
    CheckBox 30, 8, 62, 15, "Show Option 2", .CheckBox1
    GroupBox 28, 34, 79, 47, "Group"
    OptionGroup .OptionGroup1
      OptionButton 41, 47, 52, 10, "Option 1", .OptionButton1
      OptionButton 41, 62, 58, 11, "Option 2", .OptionButton2
    End Dialog
  Dim dlg As newdlg
  button = Dialog(dlg)
End Sub

Function FileDlgFunction(identifier$, action, suppvale)
  Select Case action
  Case 1
    DlgVisible 6,0
  Case 2 'user changed control or clicked a button
    If DlgControlID(identifier$)=2 then
      If DlgVisible(6)<>1 then
        DlgVisible 6
      End If
    End If
  End Select
End Function

```

See Also

| | |
|--|--------------------------------------|
| <code>BeginDialog</code> | <code>DlgFocus</code> function |
| <code>End Dialog</code> | <code>DlgFocus</code> statement |
| <code>DlgControlID</code> function | <code>DlgSetPicture</code> statement |
| <code>DlgEnable</code> function | <code>DlgText</code> function |
| <code>DlgEnable</code> statement | <code>DlgText</code> statement |
| <code>DlgListBoxArray</code> function | <code>DlgValue</code> function |
| <code>DlgListBoxArray</code> statement | <code>DlgVisible</code> function |

Do...Loop

Do...Loop Statement

Description Repeats a series of program lines as long as (or until) an expression is TRUE.

Syntax

Syntax A **Do** [{While | Until} *condition*]
 [*statement_block*]
 [Exit Do]
 [*statement_block*]
 Loop

Syntax B **Do**
 [*statement_block*]
 [Exit Do]
 [*statement_block*]
 Loop [{While | Until} *condition*]

| Syntax Element | Description |
|---------------------------|--|
| <i>Condition</i> | Any expression that evaluates to TRUE (nonzero) or FALSE (0). |
| <i>statement_block(s)</i> | Program lines to repeat while (or until) <i>condition</i> is TRUE. |

Comments When an `Exit Do` statement is executed, control goes to the statement after the `Loop` statement. When used within a nested loop, an `Exit Do` statement moves control out of the immediately enclosing loop.

Example This example lists the contents of the diskette in drive A.

```
Sub main
Dim msgret
Dim directory, count
Dim x, msgtext
Dim A()
msgret=MsgBox("Insert a disk in drive A.")
count=1
ReDim A(100)
directory=Dir ("A:\*.*")
Do While directory<>""
  A(count)=directory
  count=count+1
  directory=Dir
Loop
msgtext="Directory of drive A:\ is:" & Chr(10)
For x=1 to count
  msgtext=msgtext & A(x) & Chr(10)
Next x
MsgBox msgtext
End Sub
```

See Also `Exit` `Stop`
 `For...Next` `While...Wend`

DoEvents

Statement

Description Yields execution to Windows for processing operating system events.

Syntax **DoEvents**

Comments `DoEvents` does not return until Windows has finished processing all events in the queue and all keys sent by the `InputKeys` statement.

`DoEvents` should not be used if other tasks can interact with the running program in unforeseen ways. Since `SQABasic` yields control to the operating system at regular intervals, `DoEvents` should only be used to force `SQABasic` to allow other applications to run at a known point in the program.

Example This example activates the Windows Phone Dialer application, dials the number, and then allows the operating system to process events.

```
Sub Main
  Dim phoneNumber, msgtext
  Dim i
  InputKeys "{LeftWin}"
  InputKeys "r"
  InputKeys "dialer.exe{enter}"
  phoneNumber=InputBox("Type telephone number to call:")
  AppActivate "Phone Dialer"
  For i = 1 to 5
    DoEvents
  Next i
  InputKeys phoneNumber + "{Enter}"
  msgtext="Dialing..."
  MsgBox msgtext
  DoEvents
End Sub
```

See Also `AppActivate`
 `InputKeys`
 `Shell`

DropComboBox

Statement

Description Creates a combination of a drop-down list box and a text box.

Syntax Syntax A **DropComboBox** *x, y, dx, dy, text\$, .field*

Syntax B **DropComboBox** *x, y, dx, dy, stringarray\$, .field*

| Syntax Element | Description |
|----------------------|---|
| <i>x, y</i> | The upper left corner coordinates of the list box, relative to the upper left corner of the dialog box. |
| <i>dx, dy</i> | The width and height of the combo box in which the user enters or selects text. |
| <i>text\$</i> | A string containing the selections for the combo box. |
| <i>stringarray\$</i> | An array of dynamic strings for the selections in the combo box. |
| <i>.field</i> | The name of the dialog-record field that will hold the text string entered in the text box or chosen from the list box. |

Comments The *x* argument is measured in 1/4 system-font character-width units. The *y* argument is measured in 1/8 system-font character-width units. (See `Begin Dialog` for more information.)

The *text\$* argument must be defined, using a `Dim` statement, before the `Begin Dialog` statement is executed. The arguments in the *text\$* string are entered as shown in the following example:

```
dimname="listchoice"+Chr$(9)+"listchoice"+Chr$(9)+"listchoice"...
```

The string in the text box will be recorded in the field designated by the *.field* argument when the OK button (or any `PushButton` other than `Cancel`) is pushed. The *field* argument is also used by the dialog statements that act on this control.

You use a drop combo box when you want the user to be able to edit the contents of the list box (such as file names or their paths). You use a drop list box when the items in the list should remain unchanged.

Use the `DropComboBox` statement only between a `Begin Dialog` and an `End Dialog` statement.

Example This example defines a dialog box with a drop combo box and the OK and Cancel buttons.

```

Sub main
  Dim cchoices as String
  On Error Resume Next
  cchoices="All"+Chr$(9)+"Nothing"
  Begin Dialog UserDialog 180, 95, "SQABasic Dialog Box"
    ButtonGroup .ButtonGroup1
    Text 9, 3, 69, 13, "Filename:", .Text1
    DropComboBox 9, 17, 111, 41, cchoices, .ComboBox1
    OKButton 131, 8, 42, 13
    CancelButton 131, 27, 42, 13
  End Dialog
  Dim mydialogbox As UserDialog
  Dialog mydialogbox
  If Err=102 then
    MsgBox "You pressed Cancel."
  Else
    MsgBox "You pressed OK."
  End If
End Sub

```

See Also

| | | |
|--------------|-------------|----------------|
| Begin Dialog | CheckBox | OptionButton |
| End Dialog | ComboBox | OptionGroup |
| Button | DropListBox | Picture |
| ButtonGroup | GroupBox | StaticComboBox |
| CancelButton | ListBox | Text |
| Caption | OKButton | TextBox |

DropListBox

Statement

Description Creates a drop-down list of choices.

Syntax Syntax A **DropListBox** *x, y, dx, dy, text\$, .field*

Syntax B **DropListBox** *x, y, dx, dy, stringarray\$(), .field*

| Syntax Element | Description |
|----------------------|---|
| <i>x, y</i> | The upper left corner coordinates of the list box, relative to the upper left corner of the dialog box. |
| <i>dx, dy</i> | The width and height of the combo box in which the user enters or selects text. |
| <i>text\$</i> | A string containing the selections for the combo box. |
| <i>stringarray\$</i> | An array of dynamic strings for the selections in the combo box. |
| <i>.field</i> | The name of the dialog-record field that will hold the text string entered in the text box or chosen from the list box. |

DropListBox

Comments

The *x* argument is measured in 1/4 system-font character-width units. The *y* argument is measured in 1/8 system-font character-width units. (See `Begin Dialog` for more information.)

The *text\$* argument must be defined, using a `Dim` statement, before the `Begin Dialog` statement is executed. The arguments in the *text\$* string are entered as shown in the following example:

```
dimname="listchoice"+Chr$(9)+"listchoice"+Chr$(9)+"listchoice"...
```

The string in the text box will be recorded in the field designated by the *.field* argument when the OK button (or any `PushButton` other than `Cancel`) is pushed. The *field* argument is also used by the dialog statements that act on this control.

A drop list box is different from a list box. The drop list box only displays its list when the user selects it; the list box also displays its entire list in the dialog box.

Use the `DropListBox` statement only between a `Begin Dialog` and an `End Dialog` statement.

Example

This example defines a dialog box with a drop list box and the OK and Cancel buttons.

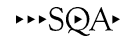
```
Sub main
  Dim DropListBox1() as String
  Dim x as Integer
  ReDim DropListBox1(3)
  For x=0 to 2
    DropListBox1(x)=Chr(65+x) & ":"
  Next x
  Begin Dialog UserDialog 186, 62, "SQABasic Dialog Box"
    Text 8, 4, 42, 8, "Drive:", .Text3
    DropListBox 8, 16, 95, 44, DropListBox1(), .DropListBox1
    OKButton 124, 6, 54, 14
    CancelButton 124, 26, 54, 14
  End Dialog
  Dim mydialog as UserDialog
  On Error Resume Next
  Dialog mydialog
  If Err=102 then
    MsgBox "Dialog box canceled."
  End If
End Sub
```

See Also

| | | |
|---------------------------|---------------------------|-----------------------------|
| <code>Begin Dialog</code> | <code>CheckBox</code> | <code>OptionGroup</code> |
| <code>End Dialog</code> | <code>ComboBox</code> | <code>Picture</code> |
| <code>Button</code> | <code>DropComboBox</code> | <code>StaticComboBox</code> |
| <code>ButtonGroup</code> | <code>Listbox</code> | <code>Text</code> |
| <code>CancelButton</code> | <code>OKButton</code> | <code>TextBox</code> |
| <code>Caption</code> | <code>OptionButton</code> | |

EditBox

User Action Command



Description Performs an action on an edit box control.

Syntax **EditBox** *action%*, *recMethod\$*, *parameters\$*

| Syntax Element | Description | | | | | | | | | | |
|------------------------|---|------------------------|-----------------------|--------------------|-----------------------|-----------------------|---------------------|-------------------|---------------------|------------------|------------------|
| <i>action%</i> | <p>One of these actions:</p> <ul style="list-style-type: none"> ▶ <i>MouseClicked</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> must contain <i>Coords=x,y</i>. ▶ <i>MouseDown</i>. The dragging of the mouse while mouse buttons and/or shifting keys (Ctrl, Alt, Shift) are pressed. When <i>action%</i> contains a mouse-drag value, <i>parameters\$</i> must contain <i>Coords=x1,y1,x2,y2</i>. See Appendix E for a list of mouse click and drag values. ▶ <i>ScrollAction</i>. One of these scroll actions: <table border="0" style="margin-left: 20px;"> <tr> <td><i>ScrollPageRight</i></td> <td><i>ScrollPageDown</i></td> </tr> <tr> <td><i>ScrollRight</i></td> <td><i>ScrollLineDown</i></td> </tr> <tr> <td><i>ScrollPageLeft</i></td> <td><i>ScrollPageUp</i></td> </tr> <tr> <td><i>ScrollLeft</i></td> <td><i>ScrollLineUp</i></td> </tr> <tr> <td><i>HScrollTo</i></td> <td><i>VScrollTo</i></td> </tr> </table> <p><i>HScrollTo</i> and <i>VScrollTo</i> take the required parameter <i>Position=%</i>.</p> | <i>ScrollPageRight</i> | <i>ScrollPageDown</i> | <i>ScrollRight</i> | <i>ScrollLineDown</i> | <i>ScrollPageLeft</i> | <i>ScrollPageUp</i> | <i>ScrollLeft</i> | <i>ScrollLineUp</i> | <i>HScrollTo</i> | <i>VScrollTo</i> |
| <i>ScrollPageRight</i> | <i>ScrollPageDown</i> | | | | | | | | | | |
| <i>ScrollRight</i> | <i>ScrollLineDown</i> | | | | | | | | | | |
| <i>ScrollPageLeft</i> | <i>ScrollPageUp</i> | | | | | | | | | | |
| <i>ScrollLeft</i> | <i>ScrollLineUp</i> | | | | | | | | | | |
| <i>HScrollTo</i> | <i>VScrollTo</i> | | | | | | | | | | |
| <i>recMethod\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <i>HTMLId=\$</i>. The text from the ID attribute of the HTML object. ▶ <i>HTMLText=\$</i>. The visible text of a Web page INPUT form element where the type is either Text or Textarea. ▶ <i>HTMLTitle=\$</i>. The text from the Title attribute of the HTML object. ▶ <i>ID=%</i>. The object's internal Windows ID. ▶ <i>Index=%</i>. The number of the object among all objects identified with the same base recognition method. Typically, <i>Index</i> is used after another recognition method qualifier — for example, <i>Name=\$; Index=%</i>. ▶ <i>JavaText=\$</i>. A label that identifies the object in the user interface. ▶ <i>Label=\$</i>. The text of the label object that immediately precedes the edit box in the Z order of windows. | | | | | | | | | | |



EditBox

▶ ▶ ▶

| Syntax Element | Description |
|---------------------|---|
| | <ul style="list-style-type: none"> ▶ Name=\$. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be Command1. ▶ ObjectIndex=%. The number of the object among all objects of the same type in the same window. ▶ State=\$. An optional qualifier for any other recognition method. There are two possible values for this setting: Enabled and Disabled. The default state is the state of the current context window (as set in the most recent Window SetContext command), or Enabled if the state has not been otherwise declared. ▶ Type=\$. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The Type qualifier uses the following form: Type=\$;recMethod=\$. Parent/child values are separated by a backslash and semicolons (;\;). ▶ VisualText=\$. An optional setting used to identify an object by its prior label. It is for user clarification only and does not affect object recognition. |
| <i>parameters\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ Coords=<i>x,y</i>. If <i>action%</i> is a mouse click, specifies the <i>x,y</i> coordinates of the click, relative to the top left of the object. ▶ Coords=<i>x1,y1,x2,y2</i>. If <i>action%</i> is a mouse drag, specifies the coordinates, where <i>x1,y1</i> are the starting coordinates of the drag, and <i>x2,y2</i> are the ending coordinates. The coordinates are relative to the top left of the object. ▶ Position=%. If <i>action%</i> is VScrollTo or HScrollTo, specifies the scroll bar value of the new scrolled-to position in the scroll box. Every scroll bar has an internal range, and this value is specific to that range. |

Comments None.

Example This example double-clicks the first edit box in the window (ObjectIndex=1) at *x,y* coordinates of 33,75.

```
EditBox DblClick, "ObjectIndex=1", "Coords=33,75"
```


This example clicks the edit box with a Name attribute of Email. The edit box is located within the Web page frame named Main.

```

EditBox Click,
    "Type=HTMLFrame;HTMLId=Main;\;Type=EditBox;Name=Email",
    "Coords=42,16"
    
```

See Also [ComboBox](#) [ComboBox](#)
 [ComboEditBox](#) [ListBox](#)

EditBoxVP

Verification Point Command



Description Establishes a verification point for an edit box control.

Syntax *Result* = **EditBoxVP** (*action%*, *recMethod\$*, *parameters\$*)

| Syntax Element | Description |
|----------------|---|
| <i>action%</i> | <p>The type of verification to perform. Valid values:</p> <ul style="list-style-type: none"> ▶ Compare. Captures the entire textual contents of the object into a grid and compares it to a recorded baseline. <i>parameters\$ VP</i> is required; <i>ExpectedResult</i> and <i>Wait</i> are optional. ▶ CompareData. Captures the contents or HTML text of the object and compares it to a recorded baseline. <i>parameters\$ VP</i> is required; <i>ExpectedResult</i> and <i>Wait</i> are optional. ▶ CompareNumeric. Captures the numeric value of the text of the object and compares it to the value of <i>parameters\$ Value or Range</i>. <i>parameters\$ VP</i> and either <i>Value or Range</i> are required; <i>ExpectedResult</i> and <i>Wait</i> are optional. ▶ CompareProperties. Captures object properties information for the object and compares it to a recorded baseline. <i>parameters\$ VP</i> is required; <i>ExpectedResult</i> and <i>Wait</i> are optional. ▶ CompareText. Captures the text of the object and compares it to a recorded baseline. <i>parameters\$ VP</i> and <i>Type</i> are required; <i>ExpectedResult</i> and <i>Wait</i> are optional. ▶ VerifyIsBlank. Checks that the object has no text. <i>parameters\$ VP</i> is required; <i>ExpectedResult</i> and <i>Wait</i> are optional. |





| Syntax Element | Description |
|----------------------|--|
| <i>recMethod</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <i>HTMLId</i>=\$. The text from the ID attribute of the HTML object. ▶ <i>HTMLText</i>=\$. The visible text of a Web page INPUT form element where the type is either Text or Textarea. ▶ <i>HTMLTitle</i>=\$. The text from the Title attribute of the HTML object. ▶ <i>ID</i>=%. The object's internal Windows ID. ▶ <i>Index</i>=%. The number of the object among all objects identified with the same base recognition method. Typically, <i>Index</i> is used after another recognition method qualifier — for example, <i>Name</i>=\$; <i>Index</i>=%. ▶ <i>JavaText</i>=\$. A label that identifies the object in the user interface. ▶ <i>Label</i>=\$. The text of the label object that immediately precedes the edit box in the internal order (Z order) of windows. ▶ <i>Name</i>=\$. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be <i>Command1</i>. ▶ <i>ObjectIndex</i>=%. The number of the object among all objects of the same type in the same window. ▶ <i>Type</i>=\$. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The <i>Type</i> qualifier uses the following form: <i>Type</i>=\$; <i>recMethod</i>=\$. Parent/child values are separated by a backslash and semicolons (<i>;</i> \ <i>;</i>). |
| <i>parameters</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <i>ExpectedResult</i>=%. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> – <i>PASS</i>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail. – <i>FAIL</i>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail. |





| Syntax Element | Description |
|----------------|--|
| | <ul style="list-style-type: none"> ▶ <code>Range=&, &</code>. Used with the action <code>CompareNumeric</code> when a numeric range comparison is being performed, as in <code>Range=2, 12</code> (test for numbers in this range). The values are inclusive. ▶ <code>Type=\$</code>. Specifies the verification method to use for <code>CompareText</code> actions. The possible values are: <code>CaseSensitive</code>, <code>CaseInsensitive</code>, <code>FindSubStr</code>, <code>FindSubStrI</code> (case insensitive), and <code>UserDefined</code>. See <i>Comments</i> for more information. If <code>UserDefined</code> is specified, two additional parameters are required: <ul style="list-style-type: none"> - <code>DLL=\$</code>. The full path and file name of the library that contains the function - <code>Function=\$</code>. The name of the custom function to use in comparing the text ▶ <code>Value=&</code>. Used with the action <code>CompareNumeric</code> when a numeric equivalence comparison is being performed, as in <code>Value=25</code> (test against the value 25). ▶ <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points. ▶ <code>Wait=%, %</code>. A <code>Wait State</code> that specifies the verification point's <code>Retry</code> value and a <code>Timeout</code> value, as in <code>Wait=10, 40</code> (retry the test every 10 seconds, but time out the test after 40 seconds). |

Comments This function returns 1 if the action performed passes or 0 if the action performed fails. See the `LogViewer` for an explanation of any failures.

With the `Type=$` parameter, `CaseSensitive` and `CaseInsensitive` require a full match between the current baseline text and the text captured during playback. With `FindSubStr` and `FindSubStrI`, the current baseline can be a substring of the text captured during playback. The substring can appear anywhere in the playback text. To modify the current baseline text, double-click the verification point name in the `Robot Asset` pane (to the left of the script).

Example This example captures the properties of the edit box identified by the label `Name` and compares them to the recorded baseline in verification point `VP TWO`. At playback, the comparison is retried every 6 seconds and times out after 30 seconds.

```
Result = EditBoxVP(CompareProperties, "Label=Name:", "VP=VPTWO;
Wait=6,30")
```

EndSaveWindowPositions

This example captures the data of the edit box with a Name attribute of Email. The edit box is located within the Web page frame named Main. `EditBoxVP` compares the data to the recorded baseline in verification point `TXTVP1`. At playback, the comparison is retried every 2 seconds and times out after 30 seconds.

```
Result = EditBoxVP (CompareData,  
    "Type=HTMLFrame;HTMLId=Main;\;Type=EditBox;Name=Email",  
    "VP=TXTVP1;Wait=2,30")
```

See Also

`LabelVP`
`PushButtonVP`
`RadioButtonVP`

EndPlay

Flow Control Command

▶▶▶SQA▶

This command is obsolete in the current version of SQABasic and should no longer be used. To maintain the upward compatibility of your existing scripts, the command does not cause an error, but it has no effect on script execution.

EndSaveWindowPositions

Utility Command

▶▶▶SQA▶

| | |
|-------------|---|
| Description | Marks the end of the script commands that save the window positions for restoration at playback. |
| Syntax | EndSaveWindowPositions |
| Comments | <p>When you record a script, Robot optionally saves the positions of all windows at the beginning of the recording. Scripts have <code>Window SetPosition</code> and <code>Window MoveTo</code> statements between <code>StartSaveWindowPositions</code> and <code>EndSaveWindowPositions</code> commands, identifying the locations and status of the windows to be restored.</p> <p><code>StartSaveWindowPositions</code> sets all playback synchronization and timeout values to zero to speed up the processing of the Window commands.</p> <p><code>EndSaveWindowPositions</code> resets all sync and timeout values to their default values.</p> <p>Script commands between <code>StartSaveWindowPositions</code> and <code>EndSaveWindowPositions</code> generate a Warning in the LogViewer if not executed properly on playback.</p> |

If you do not want to store the window position information, you can turn off this feature in the Recording Options dialog box.

On playback, the Unexpected Active Window checking is turned off between the StartSaveWindowPositions and EndSaveWindowPositions commands.

Example This example marks the end of the script commands that save the window positions for restoration at playback.

```
StartSaveWindowPositions
Window SetPosition, "Caption=TEXT.DOC",
"Coords=21,408,36,36;Status=MINIMIZED"
Window SetPosition, "Caption=Program Manager",
"Coords=-4,-4,648,488;Status=MAXIMIZED"
EndSaveWindowPositions
```

See Also StartSaveWindowPositions
Window (Actions - SetPosition and MoveTo)

Environ Function

Description Returns the string setting for a keyword in the operating system's environment table.

Syntax
 Syntax A **Environ**[\$](*environment-string*\$)
 Syntax B **Environ**[\$](*numeric expression*%)

| Syntax Element | Description |
|------------------------------|--|
| \$ | Optional. If specified the return type is String. If omitted the function will return a Variant of VarType 8 (String). |
| <i>environment-string</i> \$ | The name of a keyword in the operating system environment. |
| <i>numeric expression</i> % | A number for the position of the string in the environment table. (1st, 2nd, 3rd, etc.) |

Eof

Comments If you use the *environment-string\$* parameter, enter it in uppercase, or `Environ` returns a null string (" "). The return value for Syntax A is the string associated with the keyword requested.

If you use the *numeric expression%* parameter, the numeric expression is automatically rounded to a whole number, if necessary. The return value for Syntax B is a string in the form `keyword=value`.

`Environ` returns a null string if the specified argument cannot be found.

Example This example lists all the strings from the operating system environment table.

```
Sub main
  Dim str1(100)
  Dim msgtext
  Dim count, x
  Dim newline
  newline=Chr(10)
  x=1
  str1(x)= Environ(x)
  Do While Environ(x)<>" "
    str1(x)= Environ(x)
    x=x+1
    str1(x)=Environ(x)
  Loop
  msgtext="The Environment Strings are:" & newline & newline
  count=x
  For x=1 to count
    msgtext=msgtext & str1(x) & newline
  Next x
  MsgBox msgtext
End Sub
```

See Also None.

Eof

Function

Description Returns the value `-1` if the end of the specified open file has been reached, `0` otherwise.

Syntax `Eof (filenumber%)`

| Syntax Element | Description |
|--------------------|---|
| <i>filenumber%</i> | An integer expression identifying the open file to use. |

Comments See the `Open` statement for more information about assigning numbers to files when they are opened.

Example This example uses the `Eof` function to read records from a Random file, using a `Get` statement. The `Eof` function keeps the `Get` statement from attempting to read beyond the end of the file. The sub procedure `CREATEFILE` creates the file `C:\TEMP001` used by the main sub procedure.

```

Declare Sub createfile()
Sub main
  Dim acctno
  Dim msgtext as String
  Dim newline as String
  newline=Chr(10)
  Call createfile
  Open "C:\temp001" For Input As #1
  msgtext="The account numbers are:" & newline
  Do While Not Eof(1)
    Input #1,acctno
    msgtext=msgtext & newline & acctno & newline
  Loop
  MsgBox msgtext
  Close #1
  Kill "C:\TEMP001"
End Sub

Sub createfile()
  Rem Put the numbers 1-10 into a file
  Dim x as Integer
  Open "C:\TEMP001" for Output as #1
  For x=1 to 10
    Write #1, x
  Next x
  Close #1
End Sub

```

See Also

| | |
|-----------------|------|
| Get | Loc |
| Input function | Lof |
| Input statement | Open |
| Line Input | |

Erase

Statement

Description Reinitializes the contents of a fixed array or frees the storage associated with a dynamic array.

Syntax **Erase** Array[, Array]

| Syntax Element | Description |
|----------------|--|
| <i>Array</i> | The name of the array variable to re-initialize. |

Erase

Comments **The effect of using Erase on the elements of a fixed array varies with the type of the element:**

| Element Type | Erase Effect |
|--------------------------|--|
| numeric | Each element set to zero. |
| variable length string | Each element set to zero length string. |
| fixed length string | Each element's string is filled with zeros. |
| Variant | Each element set to Empty. |
| <i>user-defined type</i> | Members of each element are cleared as if the members were array elements, i.e. numeric members have their value set to zero, etc. |
| <i>object</i> | Each element is set to the special value Nothing. |

Example **This example prompts for a list of item numbers to put into an array and clears array if the user wants to start over.**

```
Sub main
  Dim msgtext
  Dim inum(100) as Integer
  Dim x, count
  Dim newline
  newline=Chr(10)
  x=1
  count=x
  inum(x)=0
  Do
    inum(x)=InputBox("Enter item #" & x & " (99=start over; 0=end):")
    If inum(x)=99 then
      Erase inum()
      x=0
    ElseIf inum(x)=0 then
      Exit Do
    End If
    x=x+1
  Loop
  count=x-1
  msgtext="You entered the following numbers:" & newline
  For x=1 to count
    msgtext=msgtext & inum(x) & newline
  Next x
  MsgBox msgtext
End Sub
```

See Also

Dim LBound
ReDim UBound

Erl

Function

| | | | | | | | | | |
|------------------------------|--|---------------------------|-----------------------|----------------------------|---------------------|-----------------------------|------------------------------------|------------------------------|--|
| Description | Returns the line number where an error was trapped. | | | | | | | | |
| Syntax | Erl | | | | | | | | |
| Comments | <p>If you use a <code>Resume</code> or <code>On Error</code> statement after <code>Erl</code>, the return value for <code>Erl</code> is reset to 0. To maintain the value of the line number returned by <code>Erl</code>, assign it to a variable.</p> <p>The value of the <code>Erl</code> function can be set indirectly through the <code>Error</code> statement.</p> | | | | | | | | |
| Example | <p>This example prints the error number using the <code>Err</code> function and the line number using the <code>Erl</code> statement if an error occurs during an attempt to open a file. Line numbers are automatically assigned, starting with 1, which is the <code>Sub</code> main statement.</p> <pre> Sub main Dim msgtext, userfile On Error GoTo Debugger msgtext="Enter the filename to use:" userfile=InputBox\$(msgtext) Open userfile For Input As #1 MsgBox "File opened for input." ' ...etc.... Close #1 done: Exit Sub Debugger: msgtext="Error number " & Err & " occurred at line: " & Erl MsgBox msgtext Resume done End Sub </pre> | | | | | | | | |
| See Also | <table> <tr> <td><code>Err</code> function</td> <td><code>On Error</code></td> </tr> <tr> <td><code>Err</code> statement</td> <td><code>Resume</code></td> </tr> <tr> <td><code>Error</code> function</td> <td>Trappable Error Codes (Appendix B)</td> </tr> <tr> <td><code>Error</code> statement</td> <td></td> </tr> </table> | <code>Err</code> function | <code>On Error</code> | <code>Err</code> statement | <code>Resume</code> | <code>Error</code> function | Trappable Error Codes (Appendix B) | <code>Error</code> statement | |
| <code>Err</code> function | <code>On Error</code> | | | | | | | | |
| <code>Err</code> statement | <code>Resume</code> | | | | | | | | |
| <code>Error</code> function | Trappable Error Codes (Appendix B) | | | | | | | | |
| <code>Error</code> statement | | | | | | | | | |

Err (Function)

Err Function

Description Returns the runtime error code for the last error trapped.

Syntax **Err**

Comments If you use a `Resume` or `On Error` statement after `Erl`, the return value for `Err` is reset to 0. To maintain the value of the line number returned by `Erl`, assign it to a variable.

The value of the `Err` function can be set directly through the `Err` statement, and indirectly through the `Error` statement.

Example This example prints the error number using the `Err` function and the line number using the `Erl` statement if an error occurs during an attempt to open a file. Line numbers are automatically assigned, starting with 1, which is the `Sub` main statement.

```
Sub main
  Dim msgtext, userfile
  On Error GoTo Debugger
  msgtext="Enter the filename to use:"
  userfile=InputBox$(msgtext)
  Open userfile For Input As #1
  MsgBox "File opened for input."
  ' ....etc....
  Close #1
done:
  Exit Sub
Debugger:
  msgtext="Error number " & Err & " occurred at line: " & Erl
  MsgBox msgtext
  Resume done
End Sub
```

See Also

| | |
|------------------------------|------------------------------------|
| <code>Erl</code> | <code>On Error</code> |
| <code>Err</code> statement | <code>Resume</code> |
| <code>Error</code> function | Trappable Error Codes (Appendix B) |
| <code>Error</code> statement | |

Err Statement

Description Sets a runtime error code.

Syntax **Err** = *n%*

| Syntax Element | Description |
|----------------|--|
| <i>n%</i> | An integer expression for the error code (between 1 and 32,767) or 0 for no runtime error. |

Comments The `Err` statement is used to send error information between procedures.

Example This example generates an error code of 10000 and displays an error message if a user does not enter a customer name when prompted for it. It uses the `Err` statement to clear any previous error codes before running the loop the first time and it also clears the error to allow the user to try again.

```
Sub main
  Dim custname as String
  On Error Resume Next
  Do
    Err=0
    custname=InputBox$("Enter customer name:")
    If custname="" then
      Error 10000
    Else
      Exit Do
    End If
  Select Case Err
    Case 10000
      MsgBox "You must enter a customer name."
    Case Else
      MsgBox "Undetermined error. Try again."
  End Select
  Loop Until custname<>""
  MsgBox "The name is: " & custname
End Sub
```

See Also

| | |
|------------------------------|------------------------------------|
| <code>Erl</code> | <code>On Error</code> |
| <code>Err</code> function | <code>Resume</code> |
| <code>Error</code> function | Trappable Error Codes (Appendix B) |
| <code>Error</code> statement | |

Error Function

Description Returns the error message that corresponds to the specified error code.

Syntax **Error**[\$] [(*errornumber%*)]

| Syntax Element | Description |
|---------------------|--|
| \$ | Optional. If specified, the return type is a String. If omitted, the function returns a Variant of VarType 8 (String). |
| <i>errornumber%</i> | An Integer between 1 and 32,767 specifying the error code. |

Comments If the argument is omitted, SQABasic returns the error message for the most recent runtime error.

If no error message is found to match the error code in *errornumber%*, an empty string (" ") is returned.

Example This example prints the error number, using the `Err` function, and the text of the error, using the `Error$` function, if an error occurs during an attempt to open a file.

```
Sub main
    Dim msgtext, userfile
    On Error GoTo Debugger
    msgtext="Enter the filename to use:"
    userfile=InputBox$(msgtext)
    Open userfile For Input As #1
    MsgBox "File opened for input."
    ' ...etc....
    Close #1
done:
    Exit Sub
Debugger:
    msgtext="Error " & Err & ": " & Error$
    MsgBox msgtext
    Resume done
End Sub
```

See Also

| | |
|------------------------------|------------------------------------|
| <code>Erl</code> | <code>On Error</code> |
| <code>Err</code> function | <code>Resume</code> |
| <code>Err</code> statement | Trappable Error Codes (Appendix B) |
| <code>Error</code> statement | |

Error Statement

Description Simulates the occurrence of an SQABasic or user-defined error.

Syntax **Error** *errornumber%*

| Syntax Element | Description |
|---------------------|---|
| <i>errornumber%</i> | An integer between 1 and 32,767 for the error code. |

Comments If an *errornumber%* is one that SQABasic already uses, the **Error** statement will simulate an occurrence of that error.

User-defined error codes should employ values greater than those used for standard SQABasic error codes. To help ensure that non-SQABasic error codes are chosen, user-defined codes should work down from 32,767.

If an **Error** statement is executed, and there is no error-handling routine enabled, SQABasic produces an error message and halts program execution. If an **Error** statement specifies an error code not used by SQABasic, the message **User-defined error is displayed.**

Example This example generates an error code of 10000 and displays an error message if a user does not enter a customer name when prompted for it.

```
Sub main
  Dim custname as String
  On Error Resume Next
  Do
    Err=0
    custname=InputBox$("Enter customer name:")
    If custname="" then
      Error 10000
    Else
      Exit Do
    End If
  Select Case Err
    Case 10000
      MsgBox "You must enter a customer name."
    Case Else
      MsgBox "Undetermined error. Try again."
  End Select
  Loop Until custname<>""
  MsgBox "The name is: " & custname
End Sub
```

See Also

| | |
|----------------|------------------------------------|
| Erl | On Error |
| Err function | Resume |
| Err statement | Trappable Error Codes (Appendix B) |
| Error function | |

Exp Function

Description Returns the value e (the base of natural logarithms) raised to a power.

Syntax **Exp** (*number*)

| Syntax Element | Description |
|----------------|------------------------------|
| <i>number</i> | The exponent value for e . |

Comments If the variable to contain the return value has a data type `Integer`, `Currency`, or `Single`, the return value is a single-precision value. If the variable has a data type of `Long`, `Variant`, or `Double`, the value returned is a double-precision number.

The constant e is approximately 2.718282.

Example This example estimates the value of a factorial of a number entered by the user. A factorial (represented as an exclamation mark, !) is the product of a number and each integer between it and the number 1. For example, 5 factorial, or 5!, is the product of $5 * 4 * 3 * 2 * 1$, or the value 120.

```
Sub main
    Dim x as Single
    Dim msgtext, PI
    Dim factorial as Double
    PI=3.14159
    i: x=InputBox("Enter an integer between 1 and 88: ")
    If x<=0 then
        Exit Sub
    ElseIf x>88 then
        MsgBox "The number you entered is too large. Try again."
        Goto i
    End If
    factorial=Sqr(2*PI*x)*(x^x/Exp(x))
    msgtext="The estimated factorial is: "
    msgtext=msgtext + Format(factorial, "Scientific")
    MsgBox msgtext
End Sub
```

See Also

| | |
|------------------|------------------|
| <code>Abs</code> | <code>Rnd</code> |
| <code>Fix</code> | <code>Sgn</code> |
| <code>Int</code> | <code>Sqr</code> |
| <code>Log</code> | |

FileAttr

Function

Description Returns the file mode or the operating system handle for the open file.

Syntax **FileAttr**(*filenumber%*, *returntype*)

| Syntax Element | Description |
|--------------------|--|
| <i>filenumber%</i> | An integer expression identifying the open file to use. |
| <i>returntype</i> | 1=Return file mode, 2=Return operating system handle The following table lists the return values and corresponding file modes if <i>returntype</i> is 1: 1 - Input 2 - Output 8 - Append |

Comments The argument *filenumber%* is the number used in the `Open` statement to open the file.

Example This example closes an open file if it is open for Input or Output. If open for Append, it writes a range of numbers to the file. The second sub procedure, `CREATEFILE`, creates the file and leaves it open.

```

Declare Sub createfile()
Sub main
  Dim filemode as Integer
  Dim attrib as Integer
  Dim x as Integer
  Call createfile
  attrib=1
  filemode=FileAttr(1,attrib)
  If filemode=1 or 2 then
    MsgBox "File was left open. Closing now."
    Close #1
  Else
    For x=11 to 15
      Write #1, x
    Next x
    Close #1
  End If
  Kill "C:\TEMP001"
End Sub

Sub createfile()
  Rem Put the numbers 1-10 into a file
  Dim x as Integer
  Open "C:\TEMP001" for Output as #1
  For x=1 to 10
    Write #1, x
  Next x
End Sub

```


See Also GetAttr
 Open
 SetAttr

FileCopy

Statement

Description Copies the contents of a file.

Syntax **FileCopy** *source\$, destination\$*

| Syntax Element | Description |
|----------------------|---|
| <i>source\$</i> | A string expression for the name (and path) of the file to copy. |
| <i>destination\$</i> | A string expression for the name (and path) of the file receiving the contents of <i>source\$</i> . |

Comments The contents of the file *source\$* are copied to the file *destination\$*. The original contents of *destination\$* are overwritten.

Wildcards (* or ?) are not allowed for either the *source\$* or *destination\$*. The *source\$* file cannot be copied if it is opened by SQABasic for anything other than Read access.

Example This example copies one file to another. Both file names are specified by the user.

```
Sub main
  Dim oldfile, newfile
  Dim msgtext as String
  On Error Resume Next
  oldfile= InputBox("Copy which file?")
  newfile= InputBox("Copy to?")
  FileCopy oldfile,newfile
  If Err<>0 then
    msgtext="Error during copy. Rerun program."
  Else
    msgtext="Copy successful."
  End If
  MsgBox msgtext
End Sub
```

See Also FileAttr Kill
 FileDateTime Name
 GetAttr

FileDateTime

Function

Description Returns the last modification date and time for the specified file.

Syntax **FileDateTime**(*pathname*\$)

| Syntax Element | Description |
|--------------------|--|
| <i>pathname</i> \$ | A string expression for the name of the file to query. |

Comments *Pathname*\$ can contain path and disk information, but cannot include wildcards (* and ?).

Example This example writes data to a file if it hasn't been saved within the last 2 minutes.

```

Sub main
  Dim tempfile
  Dim filetime, curtime
  Dim msgtext
  Dim acctno(100) as Single
  Dim x, I
  tempfile="C:\TEMP001"
  Open tempfile For Output As #1
  filetime=FileDateTime(tempfile)
  x=1
  I=1
  acctno(x)=0
  Do
    curtime=Time
    acctno(x)=InputBox("Enter an account number (99 to end):")
    If acctno(x)=99 then
      For I=1 to x-1
        Write #1, acctno(I)
      Next I
      Exit Do
    ElseIf (Minute(filetime)+2)<=Minute(curtime) then
      For I=I to x
        Write #1, acctno(I)
      Next I
    End If
    x=x+1
  Loop
  Close #1
  x=1
  msgtext="Contents of C:\TEMP001 is:" & Chr(10)
  Open tempfile for Input as #1
  Do While Eof(1)<>-1
    Input #1, acctno(x)
    msgtext=msgtext & Chr(10) & acctno(x)
    x=x+1
  Loop
  MsgBox msgtext
  Close #1
  Kill "C:\TEMP001"
End Sub

```

See Also FileLen
 GetAttr

FileLen

Function

Description Returns the length of the specified file.

Syntax **FileLen**(*pathname\$*)

| Syntax Element | Description |
|-------------------|--|
| <i>pathname\$</i> | A string expression that contains the name of the file to query. |

Comments *Pathname\$* can contain path and disk information, but cannot include wildcards (* and ?).

If the specified file is open, this function returns the length of the file before the file was opened.

Example This example returns the length of a file.

```
Sub main
  Dim length as Long
  Dim userfile as String
  Dim msgtext
  On Error Resume Next
  msgtext="Enter a filename:"
  userfile=InputBox(msgtext)
  length=FileLen(userfile)
  If Err<>0 then
    msgtext="Error occurred.  Rerun program."
  Else
    msgtext="The length of " & userfile & " is: " & length
  End If
  MsgBox msgtext
End Sub
```

See Also FileDateTime GetAttr
 FileLen Lof

FileVP

Verification Point Command



Description Establishes a verification point for a file or files. Tests for the existence of a file or compares two different files.

Syntax `Result = FileVP (action%, recMethod$, parameters$)`

| Syntax Element | Description |
|---------------------------|---|
| <code>action%</code> | <p>The type of verification to perform. Valid values:</p> <ul style="list-style-type: none"> ▶ Compare. Performs a binary comparison of two specified files. <code>recMethod\$</code> File1 and File2 are required. Also, <code>parameters\$</code> VP is required; <code>ExpectedResult</code> and <code>Wait</code> are optional. ▶ Exists. Checks whether a specified file exists at playback. <code>recMethod\$</code> Name is required. Also, <code>parameters\$</code> VP is required; <code>ExpectedResult</code> and <code>Wait</code> are optional. |
| <code>recMethod\$</code> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <code>File1=\$;File2=\$</code>. The full path and file names of the two files that should be compared for the action <code>Compare</code>. ▶ <code>Name=\$</code>. Name specifies the full path and file name of the file that should be tested for the action <code>Exists</code>. |
| <code>parameters\$</code> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <code>ExpectedResult=%</code>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> – <code>PASS</code>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail. – <code>FAIL</code>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail. ▶ <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points. ▶ <code>Wait=%,%</code>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10,40</code> (retry the test every 10 seconds, but time out the test after 40 seconds). |

Comments The file names specified in *recMethod\$* should contain a fully qualified path. If a drive and path are not specified, Robot uses the current directory, which is undetermined at the time of playback and will likely vary depending upon the environment and application being tested.

Verification points established through `FileVP` are not stored in the repository and do not appear in Robot's Asset pane.

Example This example tests for the existence of the MYPROG.INI file, located in the C:\WINDOWS directory. At playback, the test is retried every 4 seconds and times out after 30 seconds.

```
Result = FileVP (Exists, "Name=C:\WINDOWS\MYPROG.INI",
  "VP=FXMYPROG;Wait=4,30")
```

This example compares the contents of the files C:\MYPROG.EXE and C:\OLDPROG.EXE.

```
Result = FileVP (Compare, "File1=C:\MYPROG.EXE;
  File2=C:\OLDPROG.EXE", "VP=FCMYPROG")
```

See Also `ModuleVP`

Fix Function

Description Returns the integer part of a number.

Syntax `Fix(number)`

| Syntax Element | Description |
|----------------|-------------------------------|
| <i>number</i> | Any valid numeric expression. |

Comments The return value's data type matches the type of the numeric expression. This includes `Variant` expressions, unless the numeric expression is a string (`VarType 8`) that evaluates to a number, in which case the data type for its return value is `VarType 5` (double). If the numeric expression is `VarType 0` (empty), the data type for the return value is `VarType 3` (long).

For both positive and negative *numbers*, `Fix` removes the fractional part of the expression and returns the integer part only. For example, `Fix (6.2)` returns 6; `Fix (-6.2)` returns -6.

For...Next

Example This example returns the integer portion of a number provided by the user.

```
Sub main
  Dim usernum
  Dim intvalue
  usernum=InputBox("Enter a number with decimal places:")
  intvalue=Fix(usernum)
  MsgBox "The integer portion of " & usernum & " is: " & intvalue
End Sub
```

See Also

| | | |
|------|-----|-----|
| Abs | Int | Sgn |
| CInt | Log | Sqr |
| Exp | Rnd | |

For...Next

Statement

Description Repeats a series of program lines a fixed number of times.

Syntax

```
For counter = start TO end [STEP increment]  
  [statement_block]  
  [Exit For]  
  [statement_block]  
Next [counter]
```

| Syntax Element | Description |
|------------------------|---|
| <i>counter</i> | A numeric variable for the loop counter. |
| <i>start</i> | The beginning value of the counter. |
| <i>end</i> | The ending value of the counter. |
| <i>increment</i> | The amount by which the counter is changed each time the loop is run. (The default is one.) |
| <i>statement_block</i> | Basic functions, statements, or methods to be executed. |

Comments The *start* and *end* values must be consistent with *increment*: If *end* is greater than *start*, *increment* must be positive. If *end* is less than *start*, *increment* must be negative. SQABasic compares the sign of (*start*-*end*) with the sign of *increment*. If the signs are the same, and *end* does not equal *start*, the For...Next loop is started. If not, the loop is omitted in its entirety.

With a For...Next loop, the program lines following the For statement are executed until the Next statement is encountered. At this point, the Step amount is added to the *counter* and compared with the final value, *end*. If the beginning and ending values are the same, the loop executes once, regardless of the Step value. Otherwise, the Step value controls the loop as described in the next table:

| Step Value | Loop Execution |
|-----------------|--|
| <i>Positive</i> | If <i>counter</i> is less than or equal to <i>end</i> , the <i>Step</i> value is added to <i>counter</i> . Control returns to the statement after the <code>For</code> statement and the process repeats. If <i>counter</i> is greater than <i>end</i> , the loop is exited; execution resumes with the statement following the <code>Next</code> statement. |
| <i>Negative</i> | The loop repeats until <i>counter</i> is less than <i>end</i> . |
| <i>zero</i> | The loop repeats indefinitely. |

Within the loop, the value of the *counter* should not be changed, as changing the *counter* will make programs more difficult to maintain and debug.

`For...Next` loops can be nested within one another. Each nested loop should be given a unique variable name as its *counter*. The `Next` statement for the inside loop must appear before the `Next` statement for the outside loop. The `Exit For` statement can be used as an alternative exit from `For...Next` loops.

If the variable is left out of a `Next` statement, the `Next` statement will match the most recent `For` statement. If a `Next` statement occurs prior to its corresponding `For` statement, SQABasic will return an error message.

Multiple consecutive `Next` statements can be merged together. If this is done, the counters must appear with the innermost counter first and the outermost counter last. For example:

```
For i = 1 To 10
  [statement_block]
  For j = 1 To 5
    [statement_block]
  Next j, I
```

Example

This example calculates the factorial of a number. A factorial (represented as an exclamation mark, !) is the product of a number and each integer between it and the number 1. For example, 5 factorial, or 5!, is the product of $5 * 4 * 3 * 2 * 1$, or the value 120.

```
Sub main
  Dim number as Integer
  Dim factorial as Double
  Dim msgtext
  Dim x as Integer
  number=InputBox("Enter an integer between 1 and 170:")
  If number<=0 then
    Exit Sub
  End If
  factorial=1
  For x=number to 2 step -1
    factorial=factorial*x
  Next x
```

Format

```
Rem If number<= 35, then its factorial is small enough
Rem to be stored as a single-precision number
  If number<35 then
    factorial=CSng(factorial)
  End If
  msgtext="The factorial of " & number & " is: " & factorial
  MsgBox msgtext
End Sub
```

See Also

Do...Loop
Exit
While...Wend

Format Function

Description Returns a formatted string of an expression based on a given format.

Syntax **Format**[\$](*expression* [, *format*])

| Syntax Element | Description |
|-------------------|---|
| \$ | Optional. If specified the return type is String. If omitted the function will return a Variant of VarType 8 (String). |
| <i>expression</i> | The value to be formatted. It can be a number, Variant, or string. |
| <i>format</i> | A string expression representing the format to use. See the tables in the Comments section for the string values you can assign to this argument. |

Comments *Format* formats the *expression* as a number, date, time, or string depending upon the *format* argument. As with any string, you must enclose the *format* argument in quotation marks (" ").

Numeric values are formatted as either numbers or date/times. If a numeric expression is supplied and the *format* argument is omitted or null, the number will be converted to a string without any special formatting.

Both numeric values and Variants can be formatted as dates. When formatting numeric values as dates, the value is interpreted according the standard Basic date encoding scheme. The base date, December 30, 1899, is represented as zero, and other dates are represented as the number of days from the base date.

Strings are formatted by transferring one character at a time from the input *expression* to the output string.

When exchanging data information with external data sources or external programs, you should use double-precision floating point numbers or data strings with at least four characters for identifying the century.

Formatting Numbers

The predefined numeric formats with their meanings are as follows:

| Format | Description |
|----------------|--|
| General Number | Display the number without thousand separator. |
| Fixed | Display the number with at least one digit to the left and at least two digits to the right of the decimal separator. |
| Standard | Display the number with thousand separator and two digits to the right of decimal separator. |
| Scientific | Display the number using standard scientific notation. |
| Currency | Display the number using a currency symbol as defined in the International section of the Control Panel. Use thousand separator and display two digits to the right of decimal separator. Enclose negative value in parentheses. |
| Percent | Multiply the number by 100 and display with a percent sign appended to the right; display two digits to the right of decimal separator. |
| TRUE/FALSE | Display FALSE for 0, TRUE for any other number. |
| Yes/No | Display No for 0, Yes for any other number. |
| On/Off | Display Off for 0, On for any other number. |

To create a user-defined numeric format, follow these guidelines:

For a simple numeric format, use one or more digit characters and (optionally) a decimal separator. The two format digit characters provided are zero (0) and number sign (#). A zero forces a corresponding digit to appear in the output; while a number sign causes a digit to appear in the output if it is significant (in the middle of the number or non-zero).

| Number | Format | Result |
|---------|----------|-----------|
| 1234.56 | # | 1235 |
| 1234.56 | ##.## | 1234.56 |
| 1234.56 | ##. | 1234.6 |
| 1234.56 | #####.## | 1234.56 |
| 1234.56 | 0000.000 | 01234.560 |
| 0.12345 | ##.## | .12 |
| 0.12345 | 0.## | 0.12 |

Format

A comma placed between digit characters in a format causes a comma to be placed between every three digits to the left of the decimal separator.

| Number | Format | Result |
|--------------|----------|----------------|
| 1234567.8901 | #,#,## | 1,234,567.89 |
| 1234567.8901 | #,#,#### | 1,234,567.8901 |

Note: Although a comma and period are used in the *format* to denote separators for thousands and decimals, the output string will contain the appropriate character, based upon the current international settings for your machine.

Numbers can be scaled either by inserting one or more commas before the decimal separator or by including a percent sign in the *format* specification. Each comma preceding the decimal separator (or after all digits if no decimal separator is supplied) will scale (divide) the number by 1000. The commas will not appear in the output string. The percent sign will cause the number to be multiplied by 100. The percent sign will appear in the output string in the same position as it appears in *format*.

| Number | Format | Result |
|--------------|----------|----------|
| 1234567.8901 | #,.# | 1234.57 |
| 1234567.8901 | #,.,#### | 1.2346 |
| 1234567.8901 | #,#,.# | 1,234.57 |
| 0.1234 | #0.00% | 12.34% |

Characters can be inserted into the output string by being included in the *format* specification. The following characters will be automatically inserted in the output string in a location matching their position in the *format* specification:

- + \$ () space : /

Any set of characters can be inserted by enclosing them in double quotes. Any single character can be inserted by preceding it with a backslash (\).

| Number | Format | Result |
|------------|-------------------|-----------------------|
| 1234567.89 | \$#,0.00 | \$1,234,567.89 |
| 1234567.89 | "TOTAL:" \$#,.#00 | TOTAL: \$1,234,567.89 |
| 1234 | \=>#,#\<=\ | =>1,234<= |

You can use the SQABasic `'$CStrings` metacommand or the `Chr` function if you need to embed quotation marks in a format specification. The character code for a quotation mark is 34.

Numbers can be formatted in scientific notation by including one of the following exponent strings in the *format* specification:

E- E+ e- e+

The exponent string should be preceded by one or more digit characters. The number of digit characters following the exponent string determines the number of exponent digits in the output. *Format* specifications containing an upper case E will result in an upper case E in the output. Those containing a lower case e will result in a lower case e in the output. A minus sign following the E will cause negative exponents in the output to be preceded by a minus sign. A plus sign in the *format* will cause a sign to always precede the exponent in the output.

| Number | Format | Result |
|------------|------------|------------|
| 1234567.89 | ###.##E-00 | 123.46E04 |
| 1234567.89 | ###.##e+ # | 123.46e+ 4 |
| 0.12345 | 0.00E-00 | 1.23E-01 |

A numeric *format* can have up to four sections, separated by semicolons. If you use only one section, it applies to all values. If you use two sections, the first section applies to positive values and zeros, the second to negative values. If you use three sections, the first applies to positive values, the second to negative values, and the third to zeros. If you include semicolons with nothing between them, the undefined section is printed using the format of the first section. The fourth section applies to Null values. If it is omitted and the input expression results in a NULL value, *Format* will return an empty string.

| Number | Format | Result |
|-------------|------------------------------|----------------|
| 1234567.89 | #,0.00;(#,0.00);"Zero";"NA" | 1,234,567.89 |
| -1234567.89 | #,0.00;(#,0.00);"Zero";"NA" | (1,234,567.89) |
| 0.0 | #,0.00;(#,0.00);"Zero";"NA#" | Zero |
| 0.0 | #,0.00;(#,0.00);;"NA" | 0.00 |
| Null | #,0.00;(#,0.00);"Zero";"NA" | NA |
| Null | "The value is: " | 0.00 |

Format

Formatting Dates and Times

As with numeric formats, there are several predefined formats for formatting dates and times:

| Format | Description |
|--------------|--|
| General Date | <p>If the number has both integer and real parts, display both date and time. (for example, 11/8/1993 1:23:45 PM); if the number has only integer part, display it as a date; if the number has only fractional part, display it as time.</p> <p>The year value is generated into the formatted output as a four-digit year.</p> |
| Long Date | <p>Display a Long Date. Long Date is defined in the International section of the Control Panel.</p> |
| Medium Date | <p>Display the date using the month abbreviation and without the day of the week. (as in 08-Nov-93).</p> <p>The year value is generated into the formatted output as a two-digit year.</p> |
| Short Date | <p>Display a Short Date. Short Date is defined in the International section of the Control Panel.</p> <p>The year value is generated into the formatted output as a four-digit year.</p> |
| Long Time | <p>Display Long Time. Long Time is defined in the International section of the Control Panel and includes hours, minutes, and seconds.</p> |
| Medium Time | <p>Do not display seconds; display hours in 12-hour format and use the AM/PM designator.</p> |
| Short Time | <p>Do not display seconds; use 24-hour format and no AM/PM designator.</p> |

When using a user-defined format for a date, the *format* specification contains a series of tokens. Each token is replaced in the output string by its appropriate value.

A complete date can be output using the following tokens:

| Token | Output |
|--------|---|
| c | The date time as if the <i>format</i> was: "dddd tttt". See the definitions below. The year value is generated into the formatted output as a four-digit year. |
| dddd | The date including the day, month, and year according to the machine's current Short Date setting. The default Short Date setting for the United States is m/d/yyyy. The year value is generated into the formatted output as a four-digit year. |
| dddddd | The date including the day, month, and year according to the machine's current Long Date setting. The default Long Date setting for the United States is mmmm dd, yyyy. |
| ttttt | The time including the hour, minute, and second using the machine's current time settings. The default time format is h:mm:ss AM/PM. |

Finer control over the output is available by including *format* tokens that deal with the individual components of the date time. These tokens are:

| Token | Output |
|-------|--|
| d | The day of the month as a one or two digit number (1-31). |
| dd | The day of the month as a two digit number (01-31). |
| ddd | The day of the week as a three letter abbreviation (Sun-Sat). |
| dddd | The day of the week without abbreviation (Sunday-Saturday). |
| w | The day of the week as a number (Sunday as 1, Saturday as 7). |
| ww | The week of the year as a number (1-53). |
| m | The month of the year or the minute of the hour as a one or two digit number. The minute will be output if the preceding token was an hour; otherwise, the month will be output. |

► ► ►

Format

► ► ►

| Token | Output |
|-------|---|
| mm | The month or the year or the minute of the hour as a two digit number. The minute will be output if the preceding token was an hour; otherwise, the month will be output. |
| mmm | The month of the year as a three letter abbreviation (Jan-Dec). |
| mmmm | The month of the year without abbreviation (January-December). |
| q | The quarter of the year as a number (1-4). |
| y | The day of the year as a number (1-366). |
| yy | The year as a two-digit number (00-99). |
| yyyy | The year as a four-digit number (100-9999). |
| h | The hour as a one or two digit number (0-23). |
| hh | The hour as a two digit number (00-23). |
| n | The minute as a one or two digit number (0-59). |
| nn | The minute as a two digit number (00-59). |
| s | The second as a one or two digit number (0-59). |
| ss | The second as a two digit number (00-59). |

By default, times will be displayed using a military (24-hour) clock. Several tokens are provided in date time *format* specifications to change this default. They all cause a 12 hour clock to be used. These are:

| Token | Output |
|-------|---|
| AM/PM | An uppercase AM with any hour before noon; an uppercase PM with any hour between noon and 11:59 PM. |
| am/pm | A lowercase am with any hour before noon; a lowercase pm with any hour between noon and 11:59 PM. |
| A/P | An uppercase A with any hour before noon; an uppercase P with any hour between noon and 11:59 PM. |
| a/p | A lowercase a with any hour before noon; a lowercase p with any hour between noon and 11:59 PM. |
| AMPM | The contents of the 1159 string (s1159) in the WIN.INI file with any hour before noon; the contents of the 2359 string (s2359) with any hour between noon and 11:59 PM. Note, ampm is equivalent to AMPM. |

Any set of characters can be inserted into the output by enclosing them in double quotes. Any single character can be inserted by preceding it with a backslash (\). See number formatting above for more details.

Formatting Strings

By default, string formatting transfers characters from left to right. The exclamation point (!), when added to the *format* specification, causes characters to be transferred from right to left.

By default, characters being transferred will not be modified. The less than (<) and the greater than (>) characters can be used to force case conversion on the transferred characters. Less than forces output characters to be in lowercase. Greater than forces output characters to be in uppercase.

Character transfer is controlled by the at sign (@) and ampersand (&) characters in the *format* specification. These operate as follows:

| Character | Interpretation |
|-----------|---|
| @ | Output a character or a space. If there is a character in the string being formatted in the position where the @ appears in the format string, display it; otherwise, display a space in that position. |
| & | Output a character or nothing. If there is a character in the string being formatted in the position where the & appears, display it; otherwise, display nothing. |

A *format* specification for strings can have one or two sections separated by a semicolon. If you use one section, the format applies to all string data. If you use two sections, the first section applies to string data, the second to Null values and zero-length strings.

Example

This example calculates the square root of 2 as a double-precision floating point value and displays it in scientific notation.

```
Sub main
  Dim value
  Dim msgtext
  value=Cdbl(Sqr(2))
  msgtext="The square root of 2 is " & Format(Value,"Scientific")
  MsgBox msgtext
End Sub
```

See Also

| | | |
|------|------|--------|
| Asc | CInt | CVar |
| CCur | CLng | CVDate |
| Cdbl | CSng | Str |
| Chr | CStr | |

FreeFile

FreeFile Function

Description Returns the lowest unused file number.

Syntax **FreeFile**

Comments The `FreeFile` function is used when you need to supply a file number and want to make sure that you are not choosing a file number that is already in use. The value returned can be used in a subsequent `Open` statement.

Example This example opens a file and assigns to it the next file number available.

```
Sub main
  Dim filenumber
  Dim filename as String
  filenumber=FreeFile
  filename=InputBox("Enter a file to open: ")
  On Error Resume Next
  Open filename For Input As filenumber
  If Err<>0 then
    MsgBox "Error loading file. Re-run program."
    Exit Sub
  End If
  MsgBox "File " & filename & " opened as number: " & filenumber
  Close #filenumber
  MsgBox "File now closed."
End Sub
```

See Also `Open`

Function...End Function Statement

Description Defines a function procedure.

Syntax `[Static] [Private] Function name [(Optional) arg [As type],...)] [As functype]`
`name = expression`
End Function

| Syntax Element | Description |
|----------------|--|
| <i>name</i> | A function name. |
| <i>arg</i> | An argument to pass to the function when it is called. Multiple arguments are separated by commas. |

▶ ▶ ▶

▶ ▶ ▶

| Syntax Element | Description |
|------------------------|---|
| <i>type</i> | The data type of an argument in <i>arg</i> . |
| <i>functype</i> | The data type of the return value. |
| <i>name=expression</i> | The expression that sets the return value for the function. |

Comments

The purpose of a function is to produce and return a single value of a specified type. Recursion is supported.

The data type of *name* determines the type of the return value. Use a type declaration character as part of the *name*, or use the `As functype` clause to specify the data type. If you don't specify a data type, the default data type `Variant` is used. When calling the function, you need not specify the type declaration character.

arg contains an argument being passed to the function. An argument is represented by a variable name. Multiple arguments are separated by commas. Note the following information about the arguments being passed:

- ▶ The data type of an argument can be specified through a type declaration character or through the `As` clause.
- ▶ Arguments of a `User-Defined` data type are declared through an `As` clause and a *type* that has previously been defined through the `Type` statement.
- ▶ If an argument is an array, use empty parentheses after the argument name. The array dimensions are not specified within the `Function` statement. All references to the array within the body of the function must have a consistent number of dimensions.
- ▶ If you declare an argument as `Optional`, a procedure can omit its value when calling the function. Only arguments with `Variant` data types can be declared as optional, and all optional arguments must appear after any required arguments in the `Function` statement. Use the function `IsMissing` to check whether an optional argument was actually sent to the function or was omitted.
- ▶ Arguments can be listed in a particular order, or they can be identified by name. See the `Call` statement for information on named arguments.

You specify the return value for the function name using the *name=expression* assignment, where *name* is the name of the function and *expression* evaluates to a return value. If omitted, the value returned is `0` for numeric functions, an empty string ("") for string functions, and `VarType 0 (Empty)` for functions that return a `Variant`.

FV

The function returns to the caller when the `End Function` statement is reached or when an `Exit Function` statement is executed.

The `Static` keyword specifies that all the variables declared within the function will retain their values as long as the program is running, regardless of the way the variables are declared.

The `Private` keyword specifies that the function will not be accessible to functions and sub procedures from other modules. Only procedures defined in the same module will have access to a `Private` function.

SQABasic procedures use the call-by-reference convention by default. This means that if the called procedure changes the value of an argument passed in *arg*, the new value will apply in the calling procedure as well. This feature should be used with great care.

Use `Sub` to define a procedure with no return value.

Example

This example declares a function that is later called by the main sub procedure. The function does nothing but set its return value to 1.

```
Declare Function SBL_exfunction()  
Sub main  
    Dim y as Integer  
    Call SBL_exfunction  
    y=SBL_exfunction  
    MsgBox "The value returned by the function is: " & y  
End Sub  
  
Function SBL_exfunction()  
    SBL_exfunction=1  
End Function
```

See Also

| | |
|-----------|-----------------|
| Call | Option Explicit |
| Dim | Static |
| Global | Sub...End Sub |
| IsMissing | |

FV

Function

| | |
|-------------|---|
| Description | Returns the future value for a constant periodic stream of cash flows as in an annuity or a loan. |
|-------------|---|

Syntax **FV** (*rate*, *nper*, *pmt*, *pv*, *due*)

| Syntax Element | Description |
|----------------|--|
| <i>rate</i> | Interest rate per period. |
| <i>nper</i> | Total number of payment periods. |
| <i>pmt</i> | Constant periodic payment per period. |
| <i>pv</i> | Present value or the initial lump sum amount paid (as in the case of an annuity) or received (as in the case of a loan). |
| <i>due</i> | An integer value for when the payments are due (0=end of each period, 1= beginning of the period). |

Comments The given interest rate is assumed constant over the life of the annuity.

If payments are on a monthly schedule and the annual percentage rate on the annuity or loan is 9%, the *rate* is 0.0075 (.0075=.09/12).

Example This example finds the future value of an annuity, based on terms specified by the user.

```

Sub main
  Dim aprate, periods
  Dim payment, annuitypv
  Dim due, futurevalue
  Dim msgtext
  annuitypv=InputBox("Enter present value of the annuity: ")
  aprate=InputBox("Enter the annual percentage rate: ")
  If aprate > 1 then
    Aprate = aprate/100
  End If
  periods=InputBox("Enter the total number of pay periods: ")
  payment=InputBox("Enter the initial amount paid to you: ")
  Rem Assume payments are made at end of month due=0
  futurevalue=FV(aprate/12,periods,-payment,- annuitypv,due)
  msgtext="The future value is: " & Format(futurevalue,"Currency")
  MsgBox msgtext
End Sub

```

See Also

| | |
|------|------|
| IPmt | PPmt |
| IRR | PV |
| NPV | Rate |
| Pmt | |

GenericObject

User Action Command

Description Performs an action on a generic object.

GenericObject *action%*, *recMethod\$*, *parameters\$*

| Syntax Element | Description |
|--------------------|--|
| <i>action%</i> | <p>One of these actions:</p> <ul style="list-style-type: none"> ▶ <i>MouseClicked</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> must contain <i>Coords=x,y</i>. ▶ <i>MouseDown</i>. The dragging of the mouse while mouse buttons and/or shifting keys (Ctrl, Alt, Shift) are pressed. When <i>action%</i> contains a mouse-drag value, <i>parameters\$</i> must contain <i>Coords=x1,y1,x2,y2</i>. See Appendix E for a list of mouse click and drag values. ▶ <i>ScrollAction</i>. One of these scroll actions: <ul style="list-style-type: none"> ScrollPageRight ScrollPageDown ScrollRight ScrollLineDown ScrollPageLeft ScrollPageUp ScrollLeft ScrollLineUp HScrollTo VScrollTo <p>HScrollTo and VScrollTo take the required parameter <i>Position=%</i>.</p> <p>If Robot cannot interpret the action being applied to a scroll bar, which happens with certain custom standalone scroll bars, it records the action as a click or drag.</p> |
| <i>recMethod\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <i>Class=\$</i>. The object's class name. <i>Class</i> and <i>ClassIndex</i> are used together as a single recognition method. ▶ <i>ClassIndex=%</i>. The index or number count of the object among all objects of the same class within a given window. <i>Class</i> and <i>ClassIndex</i> are used together as a single recognition method. ▶ <i>ID=%</i>. The object's internal Windows ID. |

▶ ▶ ▶

▶ ▶ ▶

| Syntax Element | Description |
|---------------------|---|
| | <ul style="list-style-type: none"> ▶ <code>Index=%</code>. The number of the object among all objects identified with the same base recognition method. Typically, <code>Index</code> is used after another recognition method qualifier — for example, <code>Name=\$; Index=%</code>. ▶ <code>JavaText=\$</code>. A label that identifies the object in the user interface. ▶ <code>Name=\$</code>. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be <code>Command1</code>. ▶ <code>ObjectIndex=%</code>. The number of the object among all objects of the same type in the same window. ▶ <code>State=\$</code>. An optional qualifier for any other recognition method. There are two possible values for this setting: <code>Enabled</code> and <code>Disabled</code>. The default state is the state of the current context window (as set in the most recent <code>Window SetContext</code> statement), or <code>Enabled</code> if the state has not been otherwise declared. ▶ <code>Text=\$</code>. The text displayed on the object. ▶ <code>Type=\$</code>. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The <code>Type</code> qualifier uses the following form: <code>Type=\$; recMethod=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>;\;</code>). ▶ <code>VisualText=\$</code>. An optional setting used to identify an object by its visible text. It is for user clarification only and does not affect object recognition. |
| <i>parameters\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <code>Coords=x,y</code>. If <i>action%</i> is a mouse click, specifies the coordinates of the click, relative to the top left of the object. ▶ <code>Coords=x1,y1,x2,y2</code>. If <i>action%</i> is a mouse drag, specifies the coordinates, where <i>x1,y1</i> are the starting coordinates of the drag, and <i>x2,y2</i> are the ending coordinates. The coordinates are relative to the top left of the object. ▶ <code>Position=%</code>. If <i>action%</i> is <code>VScrollTo</code> or <code>HScrollTo</code>, specifies the scroll bar value of the new scrolled-to position in the scroll box. Every scroll bar has an internal range, and this value is specific to that range. |

GenericObjectVP

Example This example double-clicks the first generic object in the window (ObjectIndex=1) at x,y coordinates of 276,329.

```
GenericObject DblClick, "ObjectIndex=1", "Coords=276,329"
```

See Also GenericObjectVP

GenericObjectVP

Verification Point Command



Description Establishes a verification point for a generic object.

Syntax *Result* = **GenericObjectVP** (*action%*, *recMethod\$*, *parameters\$*)

| Syntax Element | Description |
|----------------|--|
| <i>action%</i> | <p>The type of verification to perform. Valid values:</p> <ul style="list-style-type: none">▶ CompareNumeric. Captures the numeric value of the text of the object and compares it to the value of <i>parameters\$ Value or Range</i>. <i>parameters\$ VP</i> and either <i>Value or Range</i> are required; <i>ExpectedResult</i> and <i>Wait</i> are optional.▶ CompareProperties. Captures object properties information for the object and compares it to a recorded baseline. <i>parameters\$ VP</i> is required; <i>ExpectedResult</i> and <i>Wait</i> are optional.▶ CompareText. Captures the text of the object and compares it to a recorded baseline. <i>parameters\$ VP</i> and <i>Type</i> are required; <i>ExpectedResult</i> and <i>Wait</i> are optional.▶ CompareVBXData. Captures the data from an OCX/ActiveX or VBX control and compares it to a recorded baseline. <i>parameters\$ VP</i> is required; <i>ExpectedResult</i> and <i>Wait</i> are optional.▶ VerifyIsBlank. Checks that the object has no text. <i>parameters\$ VP</i> is required; <i>ExpectedResult</i> and <i>Wait</i> are optional. |



▶ ▶ ▶

| Syntax Element | Description |
|----------------------|---|
| <i>recMethod</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ Class=\$. The object's class name. Class and ClassIndex are used together as a single recognition method. ▶ ClassIndex=%. The index or number count of the object among all objects of the same class within a given window. Class and ClassIndex are used together as a single recognition method. ▶ ID=%. The object's internal Windows ID. ▶ Index=%. The number of the object among all objects identified with the same base recognition method. Typically, Index is used after another recognition method qualifier — for example, Name=\$; Index=%. ▶ JavaText=\$. A label that identifies the object in the user interface. ▶ Name=\$. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be Command1. ▶ ObjectIndex=%. The number of the object among all objects of the same type in the same window. ▶ Text=\$. The text displayed on the object. ▶ Type=\$. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The Type qualifier uses the following form: Type=\$; recMethod=\$. Parent/child values are separated by a backslash and semicolons (; \ ;). |
| <i>parameters</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ ExpectedResult=%. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> – PASS. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail. – FAIL. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail. |

▶ ▶ ▶

GenericObjectVP

▶ ▶ ▶

| Syntax Element | Description |
|----------------|---|
| | <ul style="list-style-type: none"> ▶ <code>Range=& , &</code>. Used with the action <code>CompareNumeric</code> when a numeric range comparison is being performed, as in <code>Range=2 , 12</code> (test for numbers in this range). The values are inclusive. ▶ <code>Type=\$</code>. Specifies the verification method to use for <code>CompareText</code> actions. The possible values are: <code>CaseSensitive</code>, <code>CaseInsensitive</code>, <code>FindSubStr</code>, <code>FindSubStrI</code> (case insensitive), and <code>UserDefined</code>. See <i>Comments</i> for more information. If <code>UserDefined</code> is specified, two additional parameters are required: <ul style="list-style-type: none"> - <code>DLL=\$</code>. The full path and file name of the library that contains the function - <code>Function=\$</code>. The name of the custom function to use in comparing the text ▶ <code>Value=&</code>. Used with the action <code>CompareNumeric</code> when a numeric equivalence comparison is being performed, as in <code>Value=25</code> (test against the value 25). ▶ <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points. ▶ <code>Wait=% , %</code>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10 , 40</code> (retry the test every 10 seconds, but time out the test after 40 seconds). |

Comments This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

With the `Type=$` parameter, `CaseSensitive` and `CaseInsensitive` require a full match between the current baseline text and the text captured during playback. With `FindSubStr` and `FindSubStrI`, the current baseline can be a substring of the text captured during playback. The substring can appear anywhere in the playback text. To modify the current baseline text, double-click the verification point name in the Robot Asset pane (to the left of the script).

Example This example captures the text of the first generic object in the window (`ObjectIndex=1`) and performs a case-sensitive comparison with the recorded baseline in verification point `NEWVP`.

```
Result = GenericObjectVP (CompareText, "ObjectIndex=1",
    "VP=NEWVP;Type=CaseSensitive")
```

See Also

| | |
|------------|---------------|
| ComboBoxVP | GenericObject |
| ComboBoxVP | ListboxVP |
| EditBoxVP | |

Get Statement

Description Reads data from a file opened in Random or Binary mode and puts it in a variable.

Syntax **Get** [#]*filenumber%*, [*recnumber&*], *varname*

| Syntax Element | Description |
|-----------------------|---|
| <i>filenumber%</i> | An integer expression identifying the open file to use. |
| <i>recnumber&</i> | A Long expression containing the number of the record (for Random mode) or the offset of the byte (for Binary mode) at which to start reading. |
| <i>varname</i> | The name of the variable into which Get reads file data. <i>Varname</i> can be any variable except Object or Array variables (single array elements can be used). |

Comments For more information about how files are numbered when they're opened, see the [Open statement](#).

recnumber& is in the range 1 to 2,147,483,647. If omitted, the next record or byte is read.

Note: The commas before and after the *recnumber&* are required, even if you do not supply a *recnumber&*.

For Random mode, the following rules apply:

- ▶ Blocks of data are read from the file in chunks whose size is equal to the size specified in the Len clause of the [Open](#) statement. If the size of *varname* is smaller than the record length, the additional data is discarded. If the size of *varname* is larger than the record length, an error occurs.
- ▶ For variable length String variables, Get reads two bytes of data that indicate the length of the string, then reads the data into *varname*.
- ▶ For Variant variables, Get reads two bytes of data that indicate the type of the Variant, then it reads the body of the Variant into *varname*. Note that Variants containing strings contain two bytes of data type information followed by two bytes of length followed by the body of the string.
- ▶ User defined types are read as if each member were read separately, except no padding occurs between elements.

Get

Files opened in Binary mode behave similarly to those opened in Random mode, except:

- ▶ Get reads variables from the disk without record padding.
- ▶ Variable length Strings that are not part of user defined types are not preceded by the two-byte string length. Instead, the number of bytes read is equal to the length of *varname*.

Example

This example opens a file for Random access, gets its contents, and closes the file again. The second sub procedure, CREATEFILE, creates the C:\TEMP001 file used by the main sub procedure.

```
Declare Sub createfile()
Sub main
  Dim acctno as String*3
  Dim newline as String
  Dim recno as Long
  Dim msgtext as String
  Call createfile
  recno=1
  newline=Chr(10)
  Open "C:\TEMP001" For Random As #1 Len=3
  msgtext="The account numbers are:" & newline
  Do Until recno=11
    Get #1,recno,acctno
    msgtext=msgtext & acctno
    recno=recno+1
  Loop
  MsgBox msgtext
  Close #1
  Kill "C:\TEMP001"
End Sub

Sub createfile()
  Rem Put the numbers 1-10 into a file
  Dim x as Integer
  Open "C:\TEMP001" for Output as #1
  For x=1 to 10
    Write #1, x
  Next x
  Close #1
End Sub
```

See Also

Open
Put
Type

GetAttr

Function

Description Returns the attributes of a file, directory, or volume label.

Syntax `GetAttr(pathname$)`

| Syntax Element | Description |
|-------------------|---|
| <i>pathname\$</i> | A String expression for the name of the file, directory, or label to query. |

Comments *Pathname\$* cannot contain wildcards (* and ?).

The file attributes returned by `GetAttr` are as follows:

| Value | Meaning |
|-------|--|
| 0 | Normal file |
| 1 | Read-only file |
| 2 | Hidden file |
| 4 | System file |
| 8 | Volume label |
| 16 | Directory |
| 32 | Archive - file has changed since last backup |

Example This example tests the attributes for a file and if it is hidden, changes it to a non-hidden file.

```
Sub main
  Dim filename as String
  Dim attribs, saveattribs as Integer
  Dim answer as Integer
  Dim archno as Integer
  Dim msgtext as String
  archno=32
  On Error Resume Next
  msgtext="Enter name of a file:"
  filename=InputBox(msgtext)
  attribs=GetAttr(filename)
  If Err<>0 then
    MsgBox "Error in filename. Re-run Program."
    Exit Sub
  End If
  saveattribs=attribs
  If attribs>= archno then
    attribs=attribs-archno
  End If
```

GetField

```
Select Case attribs
  Case 2,3,6,7
    msgtext=" File: " &filename & " is hidden." & Chr(10)
    msgtext=msgtext & Chr(10) & " Change it?"
    answer=MsgBox(msgtext,308)
    If answer=6 then
      SetAttr filename, saveattribs-2
      MsgBox "File is no longer hidden."
      Exit Sub
    End If
    MsgBox "Hidden file not changed."
  Case Else
    MsgBox "File was not hidden."
End Select
End Sub
```

See Also [FileAttr](#)
[SetAttr](#)

GetField

Function

»»SQAS»

Description Returns a substring from a source string.

Syntax **GetField**[\$](*string*\$, *field_number*%, *separator_chars*\$)

| Syntax Element | Description |
|---------------------------|--|
| \$ | Optional. If specified, the return type is <code>String</code> . If omitted, the function typically returns a Variant of <code>VarType 8 (String)</code> . |
| <i>string</i> \$ | A list of fields, divided by separator characters. |
| <i>field_number</i> % | The number of the field to return, starting with 1. |
| <i>separator_chars</i> \$ | The characters separating each field. |

Comments Multiple separator characters can be specified. If *field_number* is greater than the number of fields in the string, an empty string (" ") is returned.

Example This example finds the third value in a string, delimited by plus signs (+).

```
Sub main
  Dim teststring,retvalue
  Dim msgtext
  teststring="9+8+7+6+5"
  retvalue=GetField(teststring,3,"+")
  MsgBox "The third field in: " & teststring & " is: " & retvalue
End Sub
```

| | | | |
|----------|--------------|---------------|----------|
| See Also | Left | Mid statement | SetField |
| | LTrim | Right | StrComp |
| | Mid function | RTrim | Trim |

GetLastVPResult

Utility Command

»»»SQA»

Description Returns the result of the last verification point to have been evaluated in the current playback session.

Syntax `Result = GetLastVPResult()`

Comments Each time a verification point is evaluated, a PASS or FAIL result is saved. This command returns, as an integer, the result of the last verification point to have been evaluated. The result is either PASS (integer value of 1) or FAIL (integer value of 0).

This command is useful for determining the result of a verification point that is executed in a nested script.

Example This example shows conditional execution of a script based on the result of the last verification point evaluated.

```

LastResult% = GetLastVPResult()
If LastResult% = PASS Then
...      '(If-Then routine)
End If

```

See Also None.

GetObject

Function

Description Returns an OLE2 object associated with the file name or the application name.

Syntax

Syntax A `GetObject(pathname)`

Syntax B `GetObject(pathname, class)`

Syntax C `GetObject(, class)`

| Syntax Element | Description |
|-----------------|--|
| <i>pathname</i> | The path and file name for the object to retrieve. |
| <i>class</i> | A string containing the class of the object. |

GetObject

Comments Use `GetObject` with the `Set` statement to assign a variable to the object for use in an SQABasic procedure. The variable used must first be dimensioned as an `Object`.

Syntax A of `GetObject` accesses an OLE2 object stored in a file. For example, the following two lines dimension the variable, `FILEOBJECT` as an `Object` and assign the object file `PAYABLES` to it. `PAYABLES` is located in the subdirectory `SPREDSHT`:

```
Dim FileObject As Object
Set FileObject = GetObject("\spredsht\payables")
```

If the application supports accessing component OLE2 objects within the file, you can append an exclamation point and a component object name to the file name, as follows:

```
Dim ComponentObject As Object
Set ComponentObject = GetObject("\spredsht\payables!R1C1:R13C9")
```

Syntax B of `GetObject` accesses an OLE2 object of a particular class that is stored in a file. `Class` uses the syntax: `appname.objtype`, where `appname` is the name of the application that provides the object, and `objtype` is the type or class of the object. For example:

```
Dim ClassObject As Object
Set ClassObject =
GetObject("\spredsht\payables", "turbosht.spreadsheet")
```

The third form of `GetObject` accesses the active OLE2 object of a particular class. For example:

```
Dim ActiveSheet As Object
SetActiveSheet = GetObject(, "turbosht.spreadsheet")
```

Example

This example displays a list of open files in the software application, `VISIO`. It uses the `GetObject` function to access `VISIO`. To see how this example works, you need to start `VISIO` and open one or more documents.

```
Sub main
    Dim visio as Object
    Dim doc as Object
    Dim msgtext as String
    Dim i as Integer, doccount as Integer

    'Initialize Visio
    Set visio = GetObject(,"visio.application") ' find Visio
    If (visio Is Nothing) then
        MsgBox "Couldn't find Visio!"
        Exit Sub
    End If
```

```

'Get # of open Visio files
doccount = visio.documents.count           'OLE2 call to Visio
If doccount=0 then
  msgtext="No open Visio documents."
Else
  msgtext="The open files are: " & Chr$(13)
  For i = 1 to doccount
    ' access Visio's document method
    Set doc=visio.documents(i)
    msgtext=msgtext & Chr$(13) & doc.name
  Next i
End If
MsgBox msgtext
End Sub

```

See Also

| | |
|--------------|--------------|
| Class List | Nothing |
| CreateObject | Object Class |
| Is | Typeof |
| New | |

Global Statement

Description Declare Global variables for use in an SQABasic program.

Syntax **Global** *variableName* [*As type*] [,*variableName* [*As type*]]...

| Syntax Element | Description | | | | | | | | | | |
|---------------------|--|---------|-------------------|------|---|--------|--------|--------|---------|----------|--|
| <i>variableName</i> | A variable name | | | | | | | | | | |
| <i>type</i> | <p>The data type of the variable. Valid values include:</p> <table> <tr> <td>Integer</td> <td>String (variable)</td> </tr> <tr> <td>Long</td> <td>String * <i>length</i> (<i>fixed</i>)</td> </tr> <tr> <td>Single</td> <td>Object</td> </tr> <tr> <td>Double</td> <td>Variant</td> </tr> <tr> <td>Currency</td> <td></td> </tr> </table> <p>In addition, you can specify any User-Defined data type, including a dialog box record.</p> | Integer | String (variable) | Long | String * <i>length</i> (<i>fixed</i>) | Single | Object | Double | Variant | Currency | |
| Integer | String (variable) | | | | | | | | | | |
| Long | String * <i>length</i> (<i>fixed</i>) | | | | | | | | | | |
| Single | Object | | | | | | | | | | |
| Double | Variant | | | | | | | | | | |
| Currency | | | | | | | | | | | |

Comments Global data is shared across all loaded modules. If an attempt is made to load a module that has a global variable declared that has a different data type than an existing global variable of the same name, the module load will fail.

Basic is a strongly typed language. All variables must be assigned a data type or they will be automatically assigned a type of Variant.

Global

If the `AS` clause is not used, the type of the global variable can be specified by using a type-declaration character as a suffix to *variableName*. The two different type-specification methods can be intermixed in a single `Global` statement (although not on the same variable).

Regardless of which mechanism you use to declare a global variable, you can choose to use or omit the type-declaration character when referring to the variable in the rest of your program. The type suffix is not considered part of the variable name.

Arrays

Arrays support all `SQABasic` data types. Arrays of arrays and dialog box records are not supported.

Array variables are declared by including a subscript list as part of the *variableName*. The syntax to use for *variableName* is:

```
Global variable([subscriptRange, ... ]) [As typeName]
```

where *subscriptRange* is of the format:

```
[startSubscript To] endSubscript
```

If *startSubscript* is not specified, 0 is used as the default. The `Option Base` statement can be used to change the default.

Both the *startSubscript* and the *endSubscript* are valid subscripts for the array. The maximum number of subscripts that can be specified in an array definition is 60.

If no *subscriptRange* is specified for an array, the array is declared as a dynamic array. In this case, the `ReDim` statement must be used to specify the dimensions of the array before the array can be used.

Numbers

Numeric variables can be declared using the `AS` clause and one of the following numeric types: `Currency`, `Integer`, `Long`, `Single`, `Double`. Numeric variables can also be declared by including a type character as a suffix to the name.

User-Defined

Variables of a user-defined type are declared by using an `AS` clause and a *type* that has been defined previously using the `Type` statement. The syntax is:

```
Global variableName As typeName
```

Variables of a user-defined type are made up of a collection of data elements called fields. These fields can be of any numeric, string, `Variant`, or other user-defined type. See `Type` for details on accessing fields within a user-defined type.

You cannot use the `Global` statement to declare a dialog box record (as you can with the `Dim` statement).

Strings

SQABasic supports two types of strings, fixed-length and dynamic. Fixed-length strings are declared with a specific length (between 1 and 32767) and cannot be changed later. Use the following syntax to declare a fixed-length string:

```
Global variableName As String*length
```

Dynamic strings have no declared length, and can vary in length from 0 to 32767. The initial length for a dynamic string is 0. Use the following syntax to declare a dynamic string:

```
Global variableName$ or
Global variableName As String
```

Variants

Declare variables as Variants when the type of the variable is not known at the start of, or might change during, the procedure. For example, a Variant is useful for holding input from a user when valid input can be either text or numbers. Use the following syntax to declare a Variant:

```
Global variableName or
Global variableName As Variant
```

Variant variables are initialized to `VarType Empty`.

Example

This example contains two sub procedures that share the variables `TOTAL` and `ACCTNO`, and the user-defined type `GRECORD`.

```
Type acctrecord
    acctno As Integer
End Type
Global acctno as Integer
Global total as Integer
Global grecord as acctrecord
Declare Sub createfile
Sub main
    Dim msgtext
    Dim newline as String
    Dim x as Integer
    newline=Chr$(10)
    Call createfile
    Open "C:\TEMP001" For Input as #1
    msgtext="The new account numbers are: " & newline
    For x=1 to total
        Input #1, grecord.acctno
        msgtext=msgtext & newline & grecord.acctno
    Next x
    MsgBox msgtext
    Close #1
    Kill "C:\TEMP001"
End Sub
```

GoTo

```
Sub createfile
  Dim x
  x=1
  grecord.acctno=1
  Open "C:\TEMP001" For Output as #1
  Do While grecord.acctno<>0
    grecord.acctno=InputBox("Enter 0 or new account #" & x & ":")
    If grecord.acctno<>0 then
      Print #1, grecord.acctno
      x=x+1
    End If
  Loop
  total=x-1
  Close #1
End Sub
```

See Also

| | |
|-------------|--------|
| Const | ReDim |
| Dim | Static |
| Option Base | Type |

GoTo Statement

Description Transfers program control to the specified label.

Syntax **GoTo** {*label*}

| Syntax Element | Description |
|----------------|---|
| <i>label</i> | A name beginning in the first column of a line of code and ending with a colon (:). |

Comments A label has the same format as any other SQABasic name. See Appendix A for more information about SQABasic labels and names.

To be recognized as a label, a name must begin in the first column of a line of code, and must be immediately followed by a colon (:). Keywords (such as command names) are reserved words and are not valid labels.

GoTo cannot be used to transfer control out of the current function or sub procedure.

Example This example displays the date for one week from the date entered by the user. If the date is invalid, the GoTo statement sends program execution back to the beginning.

```
Sub main
  Dim str1 as String
  Dim answer as Integer
  Dim nextweek, msgtext as String
```

```

i: str1=InputBox$("Enter a date:")
  answer=IsDate(str1)
  If answer=-1 then
    str1=CDate(str1)
    nextweek=DateValue(str1)+7
    msgtext="One week from the date entered is:"
    msgtext=msgtext & Format(nextweek,"dddddd")
    MsgBox msgtext
  Else
    MsgBox "Invalid date or format. Try again."
    GoTo i
  End If
End Sub

```

See Also Do...Loop Select Case
 For...Next While...Wend
 If...Then...Else

GroupBox Statement



Description Defines and draws a box that encloses sets of dialog box items, such as option boxes and check boxes.

Syntax **GroupBox** *x, y, dx, dy, text\$[, .id]*

| Syntax Element | Description |
|----------------|--|
| <i>x, y</i> | The upper left corner coordinates of the group box, relative to the upper left corner of the dialog box. |
| <i>dx, dy</i> | The width and height of the group box. |
| <i>text\$</i> | A string containing the title for the top border of the group box. |
| <i>.id</i> | The optional string ID for the group box, used by the dialog statements that act on this control. |

Comments The *x* argument is measured in 1/4 system-font character-width units. The *y* argument is measured in 1/8 system-font character-width units. (See *Begin Dialog* for more information.)

If *text\$* is wider than *dx*, the additional characters are truncated. If *text\$* is an empty string (" "), the top border of the group box will be a solid line.

Use the `GroupBox` statement only between a `Begin Dialog` and an `End Dialog` statement.

GroupBox (User Action Command)

Example This example creates a dialog box with two group boxes.

```
Sub main
  Begin Dialog UserDialog 242, 146, "Print Dialog Box"
    '$CStrings Save
    GroupBox 115, 14, 85, 57, "Page Range"
    OptionGroup .OptionGroup2
      OptionButton 123, 30, 46, 12, "All Pages", .OptionButton1
      OptionButton 123, 50, 67, 8, "Current Page", .OptionButton2
    GroupBox 14, 12, 85, 76, "Include"
    CheckBox 26, 17, 54, 25, "Pictures", .CheckBox1
    CheckBox 26, 36, 54, 25, "Links", .CheckBox2
    CheckBox 26, 58, 63, 25, "Header/Footer", .CheckBox3
    PushButton 34, 115, 54, 14, "Print"
    PushButton 136, 115, 54, 14, "Cancel"
    '$CStrings Restore
  End Dialog
  Dim mydialog as UserDialog
  Dialog mydialog
End Sub
```

| | | | |
|----------|--------------|--------------|----------------|
| See Also | Begin Dialog | CheckBox | OptionButton |
| | End Dialog | ComboBox | OptionGroup |
| | Button | Dialog | Picture |
| | ButtonGroup | DropComboBox | StaticComboBox |
| | CancelButton | ListBox | Text |
| | Caption | OKButton | TextBox |

GroupBox

User Action Command



Description Performs an action on a group box control.

Syntax **GroupBox** *action%*, *recMethod\$*, *parameters\$*

| Syntax Element | Description |
|----------------|---|
| <i>action%</i> | One of these mouse actions: <ul style="list-style-type: none">▶ <i>MouseClicked</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> must contain <i>Coords=x,y</i>.▶ <i>MouseDown</i>. The dragging of the mouse while mouse buttons and/or shifting keys (Ctrl, Alt, Shift) are pressed. When <i>action%</i> contains a mouse-drag value, <i>parameters\$</i> must contain <i>Coords=x1,y1,x2,y2</i>. See Appendix E for a list of mouse click and drag values. |



GroupBox (User Action Command)

▶ ▶ ▶

| Syntax Element | Description |
|----------------------|--|
| <i>recMethod</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ ID=% . The object's internal Windows ID. ▶ Name=\$. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be Command1. ▶ ObjectIndex=% . The number of the object among all objects of the same type in the same window. ▶ State=\$. An optional qualifier for any other recognition method. There are two possible values for this setting: Enabled and Disabled. The default state is the state of the current context window (as set in the most recent Window SetContext command), or Enabled if the state has not been otherwise declared. ▶ Text=\$. The text displayed on the object. ▶ VisualText=\$. An optional setting used to identify an object by its visible text. It is for user clarification only and does not affect object recognition. |
| <i>parameters</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ Coords=<i>x,y</i> . If <i>action%</i> is a mouse click, specifies the coordinates of the click, relative to the top left of the object. ▶ Coords=<i>x1,y1,x2,y2</i> . If <i>action%</i> is a mouse drag, specifies the coordinates, where <i>x1,y1</i> are the starting coordinates of the drag, and <i>x2,y2</i> are the ending coordinates. The coordinates are relative to the top left of the object. |

Comments None.

Example This example clicks the group box identified by the text CountryCodes at *x,y* coordinates of 306,223.

```
GroupBox Click, "Text=CountryCodes", "Coords=306,223"
```

See Also ComboBox EditBox
 ComboBox ListBox

GroupBoxVP

Verification Point Command

Description Establishes a verification point for a group box control.

Syntax `Result = GroupBoxVP (action%, recMethod$, parameters$)`

| Syntax Element | Description |
|--------------------|---|
| <i>action%</i> | <p>The type of verification to perform. Valid values:</p> <ul style="list-style-type: none"> ▶ CompareNumeric. Captures the numeric value of the text of the object and compares it to the value of <i>parameters\$</i> Value or Range. <i>parameters\$</i> VP and either Value or Range are required; ExpectedResult and Wait are optional. ▶ CompareProperties. Captures object properties information for the object and compares it to a recorded baseline. <i>parameters\$</i> VP is required; ExpectedResult and Wait are optional. ▶ CompareText. Captures the text of the object and compares it to a recorded baseline. <i>parameters\$</i> VP and Type are required; ExpectedResult and Wait are optional. ▶ VerifyIsBlank. Checks that the object has no text. <i>parameters\$</i> VP is required; ExpectedResult and Wait are optional. |
| <i>recMethod\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ ID=%. The object's internal Windows ID. ▶ Name=\$. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be Command1. ▶ ObjectIndex=%. The number of the object among all objects of the same type in the same window. ▶ Text=\$. The text displayed on the object. |

▶ ▶ ▶



| Syntax Element | Description |
|---------------------------|---|
| <code>parameters\$</code> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <code>ExpectedResult=%</code>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> - <code>PASS</code>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail. - <code>FAIL</code>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail. ▶ <code>Range=& , &</code>. Used with the action <code>CompareNumeric</code> when a numeric range comparison is being performed, as in <code>Range=2 , 12</code> (test for numbers in this range). The values are inclusive. ▶ <code>Type=\$</code>. Specifies the verification method to use for <code>CompareText</code> actions. The possible values are: <code>CaseSensitive</code>, <code>CaseInsensitive</code>, <code>FindSubStr</code>, <code>FindSubStrI</code> (case insensitive), and <code>UserDefined</code>. See <i>Comments</i> for more information. If <code>UserDefined</code> is specified, two additional parameters are required: <ul style="list-style-type: none"> - <code>DLL=\$</code>. The full path and file name of the library that contains the function - <code>Function=\$</code>. The name of the custom function to use in comparing the text ▶ <code>Value=&</code>. Used with the action <code>CompareNumeric</code> when a numeric equivalence comparison is being performed, as in <code>Value=25</code> (test against the value 25). ▶ <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points. ▶ <code>Wait=% , %</code>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10 , 40</code> (retry the test every 10 seconds, but time out the test after 40 seconds). |

Comments This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

Header

With the `Type=$` parameter, `CaseSensitive` and `CaseInsensitive` require a full match between the current baseline text and the text captured during playback. With `FindSubStr` and `FindSubStrI`, the current baseline can be a substring of the text captured during playback. The substring can appear anywhere in the playback text. To modify the current baseline text, double-click the verification point name in the Robot Asset pane (to the left of the script).

Example

This example captures the properties of the group box identified by the text `Icons` and compares them to the recorded baseline in verification point `GRPVP`.

```
Result = GroupBoxVP (CompareProperties, "Text=Icons", "VP=GRPVP")
```

See Also

| | |
|-------------------------|------------------------|
| <code>ComboBoxVP</code> | <code>EditBoxVP</code> |
| <code>ComboBoxVP</code> | <code>ListBoxVP</code> |

Header

User Action Command

»»SQA»

Description Performs an action on a header control.

Syntax **Header** *action%*, *recMethod\$*, *parameters\$*

| Syntax Element | Description |
|--------------------|---|
| <i>action%</i> | One of these mouse actions: <ul style="list-style-type: none">▶ <i>MouseClicked</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> must contain <code>Coords=x,y</code>.▶ <i>MouseDown</i>. The dragging of the mouse while mouse buttons and/or shifting keys (Ctrl, Alt, Shift) are pressed. When <i>action%</i> contains a mouse-drag value, <i>parameters\$</i> must contain <code>Coords=x1,y1,x2,y2</code>. See Appendix E for a list of mouse click and drag values. |
| <i>recMethod\$</i> | Valid values: <ul style="list-style-type: none">▶ <code>ID=%</code>. The object's internal Windows ID.▶ <code>Name=\$</code>. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be <code>Command1</code>. |

▶ ▶ ▶

▶ ▶ ▶

| Syntax Element | Description |
|---------------------|--|
| | <ul style="list-style-type: none"> ▶ <code>ObjectIndex=%</code>. The number of the object among all objects of the same type in the same window. ▶ <code>State=\$</code>. An optional qualifier for any other recognition method. There are two possible values for this setting: <code>Enabled</code> and <code>Disabled</code>. The default state is the state of the current context window (as set in the most recent <code>Window SetContext</code> command), or <code>Enabled</code> if the state has not been otherwise declared. ▶ <code>Text=\$</code>. The text displayed on the object. ▶ <code>VisualText=\$</code>. An optional setting used to identify an object by its visible text. It is for user clarification only and does not affect object recognition. |
| <i>parameters\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <code>Coords=x1,y1</code>. If <i>action%</i> is a mouse click, specifies the <i>x,y</i> coordinates of the click, relative to the top left of the object. ▶ <code>Coords=x1,x2,y1,y2</code>. If <i>action%</i> is a mouse drag, specifies the coordinates, where <i>x1,y1</i> are the starting coordinates of the drag, and <i>x2,y2</i> are the ending coordinates. The coordinates are relative to the top left of the object. |

Comments None.

Example This example clicks the first header control in the window (`ObjectIndex=1`) at *x,y* coordinates of `50,25`.

```
Header Click, "ObjectIndex=1", "Coords=50,25"
```

See Also `HeaderVP`

HeaderVP

Verification Point Command

▶▶SQA▶

Description Establishes a verification point for a header control.

Syntax `Result = HeaderVP (action%, recMethod$, parameters$)`

| Syntax Element | Description |
|--------------------------|--|
| <code>action%</code> | <p>The type of verification to perform. Valid values:</p> <ul style="list-style-type: none"> ▶ <code>CompareData</code>. Captures the data of the object and compares it to a recorded baseline. <code>parameters\$ VP</code> and <code>Type</code> are required; <code>ExpectedResult</code> and <code>Wait</code> are optional. ▶ <code>CompareNumeric</code>. Captures the numeric value of the text of the object and compares it to the value of <code>parameters\$ Value or Range</code>. <code>parameters\$ VP</code> and either <code>Value or Range</code> are required; <code>ExpectedResult</code> and <code>Wait</code> are optional. ▶ <code>CompareProperties</code>. Captures object properties information for the object and compares it to a recorded baseline. <code>parameters\$ VP</code> is required; <code>ExpectedResult</code> and <code>Wait</code> are optional. ▶ <code>CompareText</code>. Captures the text of the object and compares it to a recorded baseline. <code>parameters\$ VP</code> and <code>Type</code> are required; <code>ExpectedResult</code> and <code>Wait</code> are optional. |
| <code>recMethod\$</code> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <code>ID=%</code>. The object's internal Windows ID. ▶ <code>Name=\$</code>. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be <code>Command1</code>. ▶ <code>ObjectIndex=%</code>. The number of the object among all objects of the same type in the same window. ▶ <code>Text=\$</code>. The text displayed on the object. |

▶ ▶ ▶



| Syntax Element | Description |
|---------------------------|---|
| <code>parameters\$</code> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <code>ExpectedResult=%</code>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> – <code>PASS</code>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail. – <code>FAIL</code>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail. ▶ <code>Range=&, &</code>. Used with the action <code>CompareNumeric</code> when a numeric range comparison is being performed, as in <code>Range=2, 12</code> (test for numbers in this range). The values are inclusive. ▶ <code>Type=\$</code>. Specifies the verification method to use for <code>CompareText</code> actions. The possible values are: <code>CaseSensitive</code>, <code>CaseInsensitive</code>, <code>FindSubStr</code>, <code>FindSubStrI</code> (case insensitive), and <code>UserDefined</code>. See <i>Comments</i> for more information. If <code>UserDefined</code> is specified, two additional parameters are required: <ul style="list-style-type: none"> – <code>DLL=\$</code>. The full path and file name of the library that contains the function – <code>Function=\$</code>. The name of the custom function to use in comparing the text ▶ <code>Value=&</code>. Used with the action <code>CompareNumeric</code> when a numeric equivalence comparison is being performed, as in <code>Value=25</code> (test against the value 25). ▶ <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points. ▶ <code>Wait=%, %</code>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10, 40</code> (retry the test every 10 seconds, but time out the test after 40 seconds). |

Comments This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

With the `Type=$` parameter, `CaseSensitive` and `CaseInsensitive` require a full match between the current baseline text and the text captured during playback. With `FindSubStr` and `FindSubStrI`, the current baseline can be a

Hex

substring of the text captured during playback. The substring can appear anywhere in the playback text. To modify the current baseline text, double-click the verification point name in the Robot Asset pane (to the left of the script).

Example This example captures the properties of the first header control in the window (ObjectIndex=1) and compares them to the recorded baseline in verification point TEST1A.

```
Result = HeaderVP (CompareProperties, "ObjectIndex=1", "VP=TEST1A")
```

See Also Header

Hex Function

Description Returns the hexadecimal representation of a number as a string.

Syntax **Hex**[\$] (*number*)

| Syntax Element | Description |
|----------------|--|
| \$ | Optional. If specified the return type is String. If omitted the function will return a Variant of VarType 8 (String). |
| <i>number</i> | Any numeric expression that evaluates to a number. |

Comments If *number* is an integer, the return string contains up to four hexadecimal digits; otherwise, the value will be converted to a Long Integer, and the string can contain up to 8 hexadecimal digits.

To represent a hexadecimal number directly, precede the hexadecimal value with &H. For example, &H10 equals decimal 16 in hexadecimal notation.

Example This example returns the hex value for a number entered by the user.

```
Sub main
  Dim usernum as Integer
  Dim hexvalue
  usernum=InputBox("Enter a number to convert to hexadecimal:")
  hexvalue=Hex(usernum)
  MsgBox "The HEX value is: " & hexvalue
End Sub
```

See Also Oct
Format

HotKeyControl

User Action Command

»»SQA»

Description Performs an action on a hot key control.

Syntax **HotKeyControl** *action%*, *recMethod\$*, *parameters\$*

| Syntax Element | Description |
|---------------------|--|
| <i>action%</i> | <p>One of these mouse actions:</p> <ul style="list-style-type: none"> ▶ <i>MouseClicked</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> must contain <i>Coords=x,y</i>. ▶ <i>MouseDown</i>. The dragging of the mouse while mouse buttons and/or shifting keys (Ctrl, Alt, Shift) are pressed. When <i>action%</i> contains a mouse-drag value, <i>parameters\$</i> must contain <i>Coords=x1,y1,x2,y2</i>. <p>See Appendix E for a list of mouse click and drag values.</p> |
| <i>recMethod\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <i>ID=%</i>. The object's internal Windows ID. ▶ <i>Label=\$</i>. The text of the label object that immediately precedes the hot key control in the Z order of windows. ▶ <i>Name=\$</i>. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be <i>Command1</i>. ▶ <i>ObjectIndex=%</i>. The number of the object among all objects of the same type in the same window. ▶ <i>State=\$</i>. An optional qualifier for any other recognition method. There are two possible values for this setting: <i>Enabled</i> and <i>Disabled</i>. The default state is the state of the current context window (as set in the most recent <i>WindowSetContext</i> command), or <i>Enabled</i> if the state has not been otherwise declared. |
| <i>parameters\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <i>Coords=x,y</i>. If <i>action%</i> is a mouse click, specifies the coordinates of the click, relative to the top left of the object. ▶ <i>Coords=x1,y1,x2,y2</i>. If <i>action%</i> is a mouse drag, specifies the coordinates, where <i>x1,y1</i> are the starting coordinates of the drag, and <i>x2,y2</i> are the ending coordinates. The coordinates are relative to the top left of the object. |

HotKeyControlVP

Comments None.

Example This example clicks the first hot key control in the window (ObjectIndex=1) at *x,y* coordinates of 50,25.

```
HotKeyControl Click, "ObjectIndex=1", "Coords=50,25"
```

See Also HotKeyControlVP

HotKeyControlVP

Verification Point Command

»»SQA»

Description Establishes a verification point for a hot key control.

Syntax *Result* = **HotKeyControlVP** (*action%*,*recMethod\$*,*parameters\$*)

| Syntax Element | Description |
|---------------------|--|
| <i>action%</i> | The type of verification to perform. Valid value: <ul style="list-style-type: none">▶ <i>CompareProperties</i>. Captures object properties information for the object and compares it to a recorded baseline. <i>parameters\$</i> VP is required; <i>ExpectedResult</i> and <i>Wait</i> are optional. |
| <i>recMethod\$</i> | Valid values: <ul style="list-style-type: none">▶ <i>ID=%</i>. The object's internal Windows ID.▶ <i>Label=\$</i>. The text of the label object that immediately precedes the hot key control in the Z order of windows.▶ <i>Name=\$</i>. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be <i>Command1</i>.▶ <i>ObjectIndex=%</i>. The number of the object among all objects of the same type in the same window. |
| <i>parameters\$</i> | Valid values: <ul style="list-style-type: none">▶ <i>ExpectedResult=%</i>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values:<ul style="list-style-type: none">– <i>PASS</i>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail.– <i>FAIL</i>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail. |

▶ ▶ ▶

▶ ▶ ▶

| Syntax Element | Description |
|----------------|---|
| | <ul style="list-style-type: none"> ▶ <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points. ▶ <code>Wait=%, %</code>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10, 40</code> (retry the test every 10 seconds, but time out the test after 40 seconds). |

Comments This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

Example This example captures the properties of the first hot key control in the window (`ObjectIndex=1`) and compares them to the recorded baseline in verification point TEST1A.

```
Result = HotKeyControlVP (CompareProperties, "ObjectIndex=1",
"VP=TEST1A")
```

See Also `HotKeyControl`

Hour Function

Description Returns the hour of day component (0-23) of a date-time value.

Syntax `Hour (time)`

| Syntax Element | Description |
|-------------------|--|
| <code>time</code> | Any numeric or string expression that can evaluate to a date and time. |

Comments `Hour` accepts any type of `time` including strings and will attempt to convert the input value to a date value.

The return value is a Variant of VarType 2 (integer). If the value of `time` is Null, a Variant of VarType 1 (null) is returned.

`Time` is a double-precision value. The numbers to the left of the decimal point denote the date and the decimal value denotes the time (from 0 to .99999). Use the `TimeValue` function to obtain the correct value for a specific time.

HTML

Example This example extracts just the time (hour, minute, and second) from a file's last modification date and time.

```
Sub main
  Dim filename as String
  Dim ftime
  Dim hr, min
  Dim sec
  Dim msgtext as String
i: msgtext="Enter a filename:"
  filename=InputBox(msgtext)
  If filename="" then
    Exit Sub
  End If
  On Error Resume Next
  ftime=FileDateTime(filename)
  If Err<>0 then
    MsgBox "Error in file name. Try again."
    Goto i:
  End If
  hr=Hour(ftime)
  min=Minute(ftime)
  sec=Second(ftime)
  MsgBox "The file's time is: " & hr & ":" & min & ":" & sec
End Sub
```

| | | | |
|----------|------------|----------------|-----------|
| See Also | DateSerial | Now | TimeValue |
| | DateValue | Second | Weekday |
| | Day | Time function | Year |
| | Minute | Time statement | |
| | Month | TimeSerial | |

HTML

User Action Command

»»SQAS»

Description Performs a mouse action on an HTML tag.

Syntax **HTML** *action%*, *recMethod\$*, *parameters\$*

| Syntax Element | Description |
|----------------|--|
| <i>action%</i> | The following mouse action: <ul style="list-style-type: none">► Click. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). <i>parameters\$</i> must contain <i>Coords=x,y</i>. |

► ► ►



| Syntax Element | Description |
|----------------------|---|
| <i>recMethod</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <code>HTMLId=\$</code>. The text from the ID attribute of the HTML object. ▶ <code>HTMLText=\$</code>. The text of a Web page. ▶ <code>HTMLTitle=\$</code>. The text from the Title attribute of the HTML object. ▶ <code>Index=%</code>. The number of the object among all objects identified with the same base recognition method. Typically, <code>Index</code> is used after another recognition method qualifier — for example, <code>Name=\$; Index=%</code>. ▶ <code>Name=\$</code>. The Name attribute that an HTML developer assigns to an object to uniquely identify the object in the development environment. ▶ <code>Type=\$</code>. An optional qualifier for recognition methods. Used to identify the object within a Frameset. The <code>Type</code> qualifier uses the form: <code>Type=\$; recMethod=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>;\</code>). ▶ <code>VisualText=\$</code>. An optional setting used to identify an object by its visible text. It is for user clarification only and does not affect object recognition. |
| <i>parameters</i> \$ | <p>Valid value:</p> <ul style="list-style-type: none"> ▶ <code>Coords=x,y</code>. If <i>action</i>% is a mouse click, specifies the coordinates of the click, relative to the top left of the object. |

Comments None.

Example This example clicks on the Web page with the ID Obj2. This page is located within the second frame of the page.

```
Browser SetFrame, "Type=HTMFrame;Index=2", ""
Browser NewPage, "", ""
HTML Click, "HTMLId=Obj2", "Coords=481,8"
```

See Also HTMLVP

HTMLVP

Verification Point Command

»»»SQA»

Description Establishes a verification point for HTML tag.

Syntax *Result* = **HTMLVP** (*action%*, *recMethod\$*, *parameters\$*)

| Syntax Element | Description |
|--------------------|---|
| <i>action%</i> | <p>The type of verification to perform. Valid values:</p> <ul style="list-style-type: none"> ▶ CompareData. Captures the data of the object and compares it to a recorded baseline. <i>parameters\$</i> VP is required; ExpectedResult and Wait are optional. ▶ CompareProperties. Captures object properties information for the object and compares it to a recorded baseline. <i>parameters\$</i> VP is required; ExpectedResult and Wait are optional. |
| <i>recMethod\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ HTMLId=\$. The text from the ID attribute of the HTML object. ▶ HTMLText=\$. The text of a Web page. ▶ HTMLTitle=\$. The text from the Title attribute of the HTML object. ▶ Index=%. The number of the object among all objects identified with the same base recognition method. Typically, Index is used after another recognition method qualifier — for example, Name=\$; Index=%. ▶ Name=\$. The Name attribute that an HTML developer assigns to an object to uniquely identify the object in the development environment. ▶ Type=\$. An optional qualifier for recognition methods. Used to identify the object within a Frameset. The Type qualifier uses the following form: Type=\$; recMethod=\$. Parent/child values are separated by a backslash and semicolons (;\). ▶ VisualText=\$. An optional setting used to identify an object by its visible text. It is for user clarification only and does not affect object recognition. |

» » »



| Syntax Element | Description |
|----------------------|---|
| <i>parameters</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <code>ExpectedResult=%</code>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> – <code>PASS</code>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail. – <code>FAIL</code>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail. ▶ <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points. ▶ <code>Wait=%,%</code>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10,40</code> (retry the test every 10 seconds, but time out the test after 40 seconds). |

Comments This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

Example This example captures data from the tag with the ID `cmdGo`. HTMLVP compares the data to the recorded baseline in verification point `WebTest2`. At playback, the comparison is retried every 2 seconds and times out after 30 seconds.

```
Window SetContext, "WindowTag=WEBBrowser", ""
Browser NewPage, "", ""
Result = HTMLVP (CompareData, "HTMLId=cmdGo",
"VP=WebTest2;Wait=2,30")
```

See Also HTML

HTMLActiveX

User Action Command

Description Performs a mouse action on ActiveX controls embedded in the page.

Syntax **HTMLActiveX** *action%*, *recMethod\$*, *parameters\$*

| Syntax Element | Description |
|---------------------|---|
| <i>action%</i> | <p>The following mouse action:</p> <ul style="list-style-type: none"> ▶ <i>MouseClick</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> must contain <i>Coords=x,y</i>. <p>See Appendix E for a list of mouse click values.</p> |
| <i>recMethod\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <i>HTMLid=\$</i>. The text from the ID attribute of the HTML object. ▶ <i>HTMLText=\$</i>. The text of a Web page. ▶ <i>HTMLTitle=\$</i>. The text from the Title attribute of the HTML object. ▶ <i>Index=%</i>. The number of the object among all objects identified with the same base recognition method. Typically, <i>Index</i> is used after another recognition method qualifier — for example, <i>Name=\$;Index=%</i>. ▶ <i>Name=\$</i>. The Name attribute that an HTML developer assigns to an object to uniquely identify the object in the development environment. ▶ <i>Type=\$</i>. An optional qualifier for recognition methods. Used to identify the object within a Frameset. The <i>Type</i> qualifier uses the following form: <i>Type=\$;recMethod=\$</i>. Parent/child values are separated by a backslash and semicolons (<i>;\</i>). ▶ <i>VisualText=\$</i>. An optional setting used to identify an object by its visible text. It is for user clarification only and does not affect object recognition. |
| <i>parameters\$</i> | <p>Valid value:</p> <ul style="list-style-type: none"> ▶ <i>Coords=x,y</i>. If <i>action%</i> is a mouse click, specifies the coordinates of the click, relative to the top left of the object. |

Comments None.

Example This example clicks on the ActiveX element with the ID of cmdGo.

```
Window SetContext, "WindowTag=WEBBROWSER", ""
HTMLActiveX Click, "HTMLId=cmdGo", "Coords=25,11"
```

See Also HTMLActiveXVP

HTMLActiveXVP

Verification Point Command

»»»SQA»

Description Establishes a verification point for an ActiveX control embedded in the page.

Syntax *Result* = **HTMLActiveXVP** (*action%*, *recMethod\$*, *parameters\$*)

| Syntax Element | Description |
|--------------------|---|
| <i>action%</i> | <p>The type of verification to perform. Valid value:</p> <ul style="list-style-type: none"> ▶ CompareProperties. Captures object properties information for the object and compares it to a recorded baseline. <i>parameters\$</i> VP is required; ExpectedResult and Wait are optional. |
| <i>recMethod\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ HTMLId=\$. The text from the ID attribute of the HTML object. ▶ HTMLText=\$. The text of a Web page. ▶ HTMLTitle=\$. The text from the Title attribute of the HTML object. ▶ Index=%. The number of the object among all objects identified with the same base recognition method. Typically, Index is used after another recognition method qualifier — for example, Name=\$; Index=%. ▶ Name=\$. The Name attribute that an HTML developer assigns to an object to uniquely identify the object in the development environment. ▶ Type=\$. An optional qualifier for recognition methods. Used to identify the object within a Frameset. The Type qualifier uses the following form: Type=\$; recMethod=\$. Parent/child values are separated by a backslash and semicolons (\;). ▶ VisualText=\$. An optional setting used to identify an object by its visible text. It is for user clarification only and does not affect object recognition. |

▶ ▶ ▶

HTMLActiveXVP

▶ ▶ ▶

| Syntax Element | Description |
|----------------------|---|
| <i>parameters</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ ExpectedResult=%. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> – PASS. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail. – FAIL. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail. ▶ VP=\$. The verification point ID. IDs must be unique within a script. Required for all verification points. ▶ Wait=%,%. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10,40</code> (retry the test every 10 seconds, but time out the test after 40 seconds). |

Comments This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

Example This example captures properties for the ActiveX with the ID cmdGo. HTMLActiveXVP compares the properties to the recorded baseline in verification point WebTest2. At playback, the comparison is retried every 2 seconds and times out after 30 seconds.

```
Window SetContext, "WindowTag=WEBBrowser", ""
Browser NewPage, "", ""
Result = HTMLActiveXVP (CompareProperties, "HTMLId=cmdGo",
"VP=WebTest2;Wait=2,30")
```

See Also HTMLActiveX

HTMLDocument

User Action Command

»»SQA»

Description Performs a mouse action on the text of a Web page. Primarily used to position the cursor.

Syntax **HTMLDocument** *action%*, *recMethod\$*, *parameters\$*

| Syntax Element | Description |
|--------------------|--|
| <i>action%</i> | <p>One of these mouse actions:</p> <ul style="list-style-type: none"> ▶ Click. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). <i>parameters\$</i> must contain <i>Coords=x,y</i>. ▶ MouseDown. The dragging of the mouse while mouse buttons and/or shifting keys (Ctrl, Alt, Shift) are pressed. When <i>action%</i> contains a mouse-drag value, <i>parameters\$</i> must contain <i>Coords=x1,y1,x2,y2</i>. <p>See Appendix E for a list of mouse click and drag values.</p> |
| <i>recMethod\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ HTMLId=\$. The text from the ID attribute of the HTML object. ▶ HTMLText=\$. The text of a Web page. ▶ HTMLTitle=\$. The text from the Title attribute of the HTML object. ▶ Index=%. The number of the object among all objects identified with the same base recognition method. Typically, <i>Index</i> is used after another recognition method qualifier — for example, <i>Name=\$;Index=%</i>. ▶ Name=\$. The Name attribute that an HTML developer assigns to an object to uniquely identify the object in the development environment. ▶ Type=\$. An optional qualifier for recognition methods. Used to identify the object within a Frameset. The <i>Type</i> qualifier uses the following form: <i>Type=\$;recMethod=\$</i>. Parent/child values are separated by a backslash and semicolons (<i>;\;</i>). ▶ VisualText=\$. An optional setting used to identify an object by its visible text. It is for user clarification only and does not affect object recognition. |

▶ ▶ ▶

HTMLDocumentVP

▶ ▶ ▶

| Syntax Element | Description |
|---------------------|--|
| <i>parameters\$</i> | <p>Valid values:</p> <ul style="list-style-type: none">▶ <i>Coords=x,y</i>. If <i>action%</i> is a mouse click, specifies the coordinates of the click, relative to the top left of the object.▶ <i>Coords=x1,y1,x2,y2</i>. If <i>action%</i> is a mouse drag, specifies the coordinates, where <i>x1,y1</i> are the starting coordinates of the drag, and <i>x2,y2</i> are the ending coordinates. The coordinates are relative to the top left of the object. |

Comments None.

Example This example clicks on the Web page with the title My Web Page. This page is located within the second frame of the page.

```
Browser SetFrame, "Type=HTMFrame;Index=2", ""
Browser NewPage, "HTMLTitle=My Web Page", ""
HTMLDocument Click, "HTMLTitle=My Web Page", "Coords=481,8"
```

See Also HTMLDocumentVP

HTMLDocumentVP

Verification Point Command

▶▶SQA▶

Description Establishes a verification point for Web page data.

Syntax *Result* = **HTMLDocumentVP** (*action%*,*recMethod\$*,*parameters\$*)

| Syntax Element | Description |
|----------------|---|
| <i>action%</i> | <p>The type of verification to perform. Valid values:</p> <ul style="list-style-type: none">▶ <i>CompareData</i>. Captures the data of the object and compares it to a recorded baseline. <i>parameters\$ VP</i> is required; <i>ExpectedResult</i> and <i>Wait</i> are optional.▶ <i>CompareProperties</i>. Captures object properties information for the object and compares it to a recorded baseline. <i>parameters\$ VP</i> is required; <i>ExpectedResult</i> and <i>Wait</i> are optional. |

▶ ▶ ▶

▶ ▶ ▶

| Syntax Element | Description |
|----------------------|--|
| <i>recMethod</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <code>HTMLId=\$</code>. The text from the ID attribute of the HTML object. ▶ <code>HTMLText=\$</code>. The text of a Web page. ▶ <code>HTMLTitle=\$</code>. The text from the Title attribute of the HTML object. ▶ <code>Index=%</code>. The number of the object among all objects identified with the same base recognition method. Typically, <code>Index</code> is used after another recognition method qualifier — for example, <code>Name=\$; Index=%</code>. ▶ <code>Name=\$</code>. The Name attribute that an HTML developer assigns to an object to uniquely identify the object in the development environment. ▶ <code>Type=\$</code>. An optional qualifier for recognition methods. Used to identify the object within a Frameset. The <code>Type</code> qualifier uses the following form: <code>Type=\$; recMethod=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>;\;</code>). ▶ <code>VisualText=\$</code>. An optional setting used to identify an object by its visible text. It is for user clarification only and does not affect object recognition. |
| <i>parameters</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <code>ExpectedResult=%</code>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> – <code>PASS</code>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail. – <code>FAIL</code>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail. ▶ <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points. ▶ <code>Wait=%, %</code>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10, 40</code> (retry the test every 10 seconds, but time out the test after 40 seconds). |

Comments This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

HTMLHiddenVP

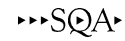
Example This example captures data from the Web page titled My Web Page. The page is located within the second frame of the page. `HTMLDocumentVP` compares the data to the recorded baseline in verification point `WebTest2`. At playback, the comparison is retried every 2 seconds and times out after 30 seconds.

```
Browser SetFrame, "Type=HTMFrame;Index=2",""  
Browser NewPage, "HTMLTitle=My Web Page",""  
Result = HTMLDocumentVP (CompareData, "HTMLTitle=My Web Page",  
"VP=WebTest2;Wait=2,30")
```

See Also [HTMLDocument](#)

HTMLHidden

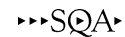
Keyword



`HTMLHidden` is an unused reserved keyword.

HTMLHiddenVP

Verification Point Command



Description Establishes a verification point for a hidden element.

Syntax `Result = HTMLHiddenVP (action%, recMethod$, parameters$)`

| Syntax Element | Description |
|--------------------------|--|
| <code>action%</code> | The type of verification to perform. Valid values: <ul style="list-style-type: none">▶ <code>CompareData</code>. Captures the data of the object and compares it to a recorded baseline. <code>parameters\$</code> VP is required; <code>ExpectedResult</code> and <code>Wait</code> are optional.▶ <code>CompareProperties</code>. Captures object properties information for the object and compares it to a recorded baseline. <code>parameters\$</code> VP is required; <code>ExpectedResult</code> and <code>Wait</code> are optional. |
| <code>recMethod\$</code> | Valid values: <ul style="list-style-type: none">▶ <code>HTMLId=\$</code>. The text from the ID attribute of the HTML object.▶ <code>HTMLText=\$</code>. The text of a Web page.▶ <code>HTMLTitle=\$</code>. The text from the Title attribute of the HTML object.▶ <code>Index=%</code>. The number of the object among all objects identified with the same base recognition method. |



▶ ▶ ▶

| Syntax Element | Description |
|---------------------------|---|
| | <p>Typically, <code>Index</code> is used after another recognition method qualifier — for example, <code>Name=\$; Index=%</code>.</p> <ul style="list-style-type: none"> ▶ <code>Name=\$</code>. The <code>Name</code> attribute that an HTML developer assigns to an object to uniquely identify the object in the development environment. ▶ <code>Type=\$</code>. An optional qualifier for recognition methods. Used to identify the object within a <code>Frameset</code>. The <code>Type</code> qualifier uses the following form: <code>Type=\$; recMethod=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>;\;</code>). ▶ <code>VisualText=\$</code>. An optional setting used to identify an object by its visible text. It is for user clarification only and does not affect object recognition. |
| <code>parameters\$</code> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <code>ExpectedResult=%</code>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> – <code>PASS</code>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail. – <code>FAIL</code>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail. ▶ <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points. ▶ <code>Wait=%, %</code>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10, 40</code> (retry the test every 10 seconds, but time out the test after 40 seconds). |

Comments This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

Example This example captures data from the hidden element with the ID `Hidden`. The element is located within the second frame of the page. `HTMLHiddenVP` compares the data to the recorded baseline in verification point `WebTest2`. At playback, the comparison is retried every 2 seconds and times out after 30 seconds.

```
Browser SetFrame, "Type=HTMFrame;Index=2", ""
Browser NewPage, "", ""
Result = HTMLHiddenVP (CompareData, "HTMLId=Hidden",
    "VP=WebTest2;Wait=2,30")
```

HTMLImage

See Also None.

HTMLImage

User Action Command

»»SQA»

Description Performs a mouse click on an image of a Web page.

Syntax **HTMLImage** *action%*, *recMethod\$*, *parameter\$*

| Syntax Element | Description |
|--------------------|--|
| <i>action%</i> | The following mouse action: ▶ Click. A mouse click on an image. |
| <i>recMethod\$</i> | Valid values: ▶ HTMLId=\$. The text from the ID attribute of the HTML object. ▶ HTMLText=\$. The text of a Web page. ▶ HTMLTitle=\$. The text from the Title attribute of the HTML object. ▶ Index=%. The number of the object among all objects identified with the same base recognition method. Typically, Index is used after another recognition method qualifier — for example, Name=\$; Index=%. ▶ Name=\$. The Name attribute that an HTML developer assigns to an object to uniquely identify the object in the development environment. ▶ Type=\$. An optional qualifier for recognition methods. Used to identify the object within a Frameset. The Type qualifier uses the following form: Type=\$; recMethod=\$. Parent/child values are separated by a backslash and semicolons (: \ ;). ▶ VisualText=\$. An optional setting used to identify an object by its visible text. It is for user clarification only and does not affect object recognition. |
| <i>parameter\$</i> | Valid values: ▶ AreaId=\$. An ID assigned to an area in an HTML image map. Used with client-side image maps. ▶ AreaIndex=%. An ID assigned to an HTML image map. The number of the area among all areas of the same type within an HTML image map. Used with client-side image maps. |

▶ ▶ ▶

▶ ▶ ▶

| Syntax Element | Description |
|----------------|---|
| | <ul style="list-style-type: none"> ▶ AreaName=\$. A name assigned to an area in an HTML image map. Used with client-side image maps. ▶ Coords=<i>x,y</i>. If <i>action%</i> is a mouse click, specifies the coordinates of the click, relative to the top left of the object. Used with server-side image maps. |

Comments This command supports both client-side and server-side image maps.

Example This example clicks the image with a Value attribute of Button. The image is located within the second frame of the page.

```
Browser SetFrame, "Type=HTMFrame;Index=2", ""
Browser NewPage, "HTMLTitle=My Web Page", ""
HTMLImage Click, "Type=HTMLImage;HTMLText=Button", "Coords=12,13"
```

See Also HTMLImageVP

HTMLImageVP

Verification Point Command

▶▶SQA▶

Description Establishes a verification point for a Web page image.

Syntax *Result* = **HTMLImageVP** (*action%*, *recMethod*\$, *parameter*\$)

| Syntax Element | Description |
|---------------------|--|
| <i>action%</i> | <p>The type of verification to perform. Valid values:</p> <ul style="list-style-type: none"> ▶ CompareData. Captures the data of the object and compares it to a recorded baseline. <i>parameters</i>\$ VP is required; ExpectedResult and Wait are optional. ▶ CompareProperties. Captures object properties information for the object and compares it to a recorded baseline. <i>parameters</i>\$ VP is required; ExpectedResult and Wait are optional. |
| <i>recMethod</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ HTMLId=\$. The text from the ID attribute of the HTML object. ▶ HTMLText=\$. The text of a Web page. ▶ HTMLTitle=\$. The text from the Title attribute of the HTML object. |

▶ ▶ ▶

HTMLImageVP

▶ ▶ ▶

| Syntax Element | Description |
|---------------------|---|
| | <ul style="list-style-type: none"> ▶ <code>Index=%</code>. The number of the object among all objects identified with the same base recognition method. Typically, <code>Index</code> is used after another recognition method qualifier — for example, <code>Name=\$; Index=%</code>. ▶ <code>Name=\$</code>. The Name attribute that an HTML developer assigns to an object to uniquely identify the object in the development environment. ▶ <code>Type=\$</code>. An optional qualifier for recognition methods. Used to identify the object within a Frameset. The <code>Type</code> qualifier uses the following form: <code>Type=\$; recMethod=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>i\i</code>). ▶ <code>VisualText=\$</code>. An optional setting used to identify an object by its visible text. It is for user clarification only and does not affect object recognition. |
| <i>parameter</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <code>ExpectedResult=%</code>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> – <code>PASS</code>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail. – <code>FAIL</code>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail. ▶ <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points. ▶ <code>Wait=%, %</code>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10, 40</code> (retry the test every 10 seconds, but time out the test after 40 seconds). |

Comments This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

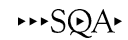
Example This example captures data from the image with the Name attribute of Red Button. The image is located within the second frame of the page. HTMLImageVP compares the data to the recorded baseline in verification point ImageData2.

```
Browser SetFrame, "Type=HTMLFrame;Index=2", ""
Browser NewPage, "HTMLTitle=My Web Page", ""
Result = HTMLImageVP (CompareData, " Type=HTMLImage;
Name=Red Button", "VP=ImageData2")
```

See Also [HTMLImage](#)

HTMLLink

User Action Command



Description Performs a mouse click on a Web page link.

Syntax **HTMLLink** *action%*, *recMethod\$*, *parameter\$*

| Syntax Element | Description |
|--------------------|---|
| <i>action%</i> | <p>The following mouse action:</p> <ul style="list-style-type: none"> ▶ Click. A mouse click on a link. |
| <i>recMethod\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ HTMLId=\$. The text from the ID attribute of the HTML object. ▶ HTMLText=\$. The text of a Web page link. ▶ HTMLTitle=\$. The text from the Title attribute of the HTML object. ▶ Index=%. The number of the object among all objects identified with the same base recognition method. Typically, Index is used after another recognition method qualifier — for example, Name=\$; Index=%. ▶ Name=\$. The Name attribute that an HTML developer assigns to an object to uniquely identify the object in the development environment. ▶ Type=\$. An optional qualifier for recognition methods. Used to identify the object within a Frameset. The Type qualifier uses the following form: Type=\$; recMethod=\$. Parent/child values are separated by a backslash and semicolons (;\;). ▶ VisualText=\$. An optional setting used to identify an object by its visible text. It is for user clarification only and does not affect object recognition. |
| <i>parameter\$</i> | <p>Valid value:</p> <ul style="list-style-type: none"> ▶ [empty quotes]. Robot performs the click based upon the recognition method. |

Comments None.

HTMLinkVP

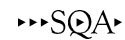
Example This example clicks on the Web page link with the text Home Page. The link is located within the second frame of the page.

```
Browser SetFrame, "Type=HTMFrame;Index=2", ""
Browser NewPage, "HTMLTitle=My Web Page", ""
HTMLink Click, "Type=HTMLink;HTMLText=Home Page", ""
```

See Also HTMLinkVP

HTMLinkVP

Verification Point Command



Description Establishes a verification point for a Web page link.

Syntax *Result* = **HTMLinkVP** (*action%*, *recMethod\$*, *parameter\$*)

| Syntax Element | Description |
|--------------------|--|
| <i>action%</i> | The type of verification to perform. Valid values: <ul style="list-style-type: none">▶ CompareData. Captures the data of the object and compares it to a recorded baseline. <i>parameters\$</i> VP is required; ExpectedResult and Wait are optional.▶ CompareProperties. Captures object properties information for the object and compares it to a recorded baseline. <i>parameters\$</i> VP is required; ExpectedResult and Wait are optional. |
| <i>recMethod\$</i> | Valid values: <ul style="list-style-type: none">▶ HTMLId=\$. The text from the ID attribute of the HTML object.▶ HTMLText=\$. The text of a Web page link.▶ HTMLTitle=\$. The text from the Title attribute of the HTML object.▶ Index=%. The number of the object among all objects identified with the same base recognition method. Typically, Index is used after another recognition method qualifier — for example, Name=\$; Index=%.▶ Name=\$. The Name attribute that an HTML developer assigns to an object to uniquely identify the object in the development environment.▶ Type=\$. An optional qualifier for recognition methods. Used to identify the object within a Frameset. The Type qualifier uses the following form: Type=\$; recMethod=\$. Parent/child values are separated by a backslash and semicolons (<i>i \ ;</i>). |



▶ ▶ ▶

| Syntax Element | Description |
|---------------------|---|
| | <ul style="list-style-type: none"> ▶ <code>VisualText=\$</code>. An optional setting used to identify an object by its visible text. It is for user clarification only and does not affect object recognition. |
| <i>parameter</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <code>ExpectedResult=%</code>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> - <code>PASS</code>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail. - <code>FAIL</code>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail. ▶ <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points. ▶ <code>Wait=%,%</code>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10,40</code> (retry the test every 10 seconds, but time out the test after 40 seconds). |

Comments This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

Example This example captures the text of the Web page link Home Page and compares the data to the recorded baseline in verification point `WebLink1`. At playback, the comparison is retried every 2 seconds and times out after 30 seconds.

```
Browser SetFrame, "Type=HTMFrame;Index=2", ""
Browser NewPage, "HTMLTitle=My Web Page", ""
Result = HTMLLinkVP (CompareData, "HTMLText=Home Page",
"VP=WebLink1";Wait=2,30)
```

See Also HTMLLink

HTMLTable

User Action Command

»»SQAS»

Description Performs a mouse action on the text of a Web page. Primarily used to position the cursor.

Syntax **HTMLDocument** *action%*, *recMethod\$*, *parameters\$*

| Syntax Element | Description |
|--------------------|--|
| <i>action%</i> | <p>One of these mouse actions:</p> <ul style="list-style-type: none"> ▶ <i>MouseClicked</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> must contain <i>Coords=x,y</i>. ▶ <i>MouseDown</i>. The dragging of the mouse while mouse buttons and/or shifting keys (Ctrl, Alt, Shift) are pressed. When <i>action%</i> contains a mouse-drag value, <i>parameters\$</i> must contain <i>Coords=x1,y1,x2,y2</i>. <p>See Appendix E for a list of mouse click and drag values.</p> |
| <i>recMethod\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <i>HTMLId=\$</i>. The text from the ID attribute of the HTML object. ▶ <i>HTMLText=\$</i>. The text of a Web page. ▶ <i>HTMLTitle=\$</i>. The caption of the table. ▶ <i>Index=%</i>. The number of the object among all objects identified with the same base recognition method. Typically, <i>Index</i> is used after another recognition method qualifier — for example, <i>Name=\$;Index=%</i>. ▶ <i>Name=\$</i>. The Name attribute that an HTML developer assigns to an object to uniquely identify the object in the development environment. ▶ <i>Type=\$</i>. An optional qualifier for recognition methods. Used to identify the object within a Frameset. The <i>Type</i> qualifier uses the following form: <i>Type=\$;recMethod=\$</i>. Parent/child values are separated by a backslash and semicolons (<i>;\;</i>). ▶ <i>VisualText=\$</i>. An optional setting used to identify an object by its visible text. It is for user clarification only and does not affect object recognition. |

▶ ▶ ▶

▶ ▶ ▶

| Syntax Element | Description |
|----------------------|--|
| <i>parameters</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ Col=%. The column number of the table. ▶ Coords=<i>x,y</i>. If <i>action</i>% is a mouse click, specifies the coordinates of the click, relative to the top left of the object. ▶ Coords=<i>x1,y1,x2,y2</i>. If <i>action</i>% is a mouse drag, specifies the coordinates, where <i>x1,y1</i> are the starting coordinates of the drag, and <i>x2,y2</i> are the ending coordinates. The coordinates are relative to the top left of the object. ▶ Row=%. The row number of the table. |

Comments None.

Example This example clicks on the Web table with the title My Table. This page is located within the second frame of the page.

```
Browser SetFrame, "Type=HTMFrame;Index=2", ""
Browser NewPage, "HTMLTitle=My Web Page", ""
HTMLTable Click, "Type=HTMLTable;HTMLTitle=My Table", "Row=6,Col=1"
```

See Also HTMLDocumentVP

HTMLTableVP

Verification Point Command

▶▶SQA▶

Description Establishes a verification point for a Web page table.

Syntax *Result* = **HTMLTableVP** (*action*%, *recMethod*\$, *parameter*\$)

| Syntax Element | Description |
|-----------------|--|
| <i>action</i> % | <p>The type of verification to perform. Valid values:</p> <ul style="list-style-type: none"> ▶ CompareData. Captures the data of the object and compares it to a recorded baseline. <i>parameters</i>\$ VP is required; ExpectedResult and Wait are optional. ▶ CompareProperties. Captures object properties information for the object and compares it to a recorded baseline. <i>parameters</i>\$ VP is required; ExpectedResult and Wait are optional. |

▶ ▶ ▶

Command Reference

6-215

▶ ▶ ▶

| Syntax Element | Description |
|---------------------|---|
| <i>recMethod</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <code>HTMLId=\$</code>. The text from the ID attribute of the HTML object. ▶ <code>HTMLText=\$</code>. The text of a Web page link. ▶ <code>HTMLTitle=\$</code>. The caption of the table. ▶ <code>Index=%</code>. The number of the object among all objects identified with the same base recognition method. Typically, <code>Index</code> is used after another recognition method qualifier — for example, <code>Name=\$; Index=%</code>. ▶ <code>Name=\$</code>. The Name attribute that an HTML developer assigns to an object to uniquely identify the object in the development environment. ▶ <code>Type=\$</code>. An optional qualifier for recognition methods. Used to identify the object within a Frameset. The <code>Type</code> qualifier uses the following form: <code>Type=\$; recMethod=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>;\;</code>). ▶ <code>VisualText=\$</code>. An optional setting used to identify an object by its visible text. It is for user clarification only and does not affect object recognition. |
| <i>parameter</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <code>ExpectedResult=%</code>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> – <code>PASS</code>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail. – <code>FAIL</code>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail. ▶ <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points. ▶ <code>Wait=%, %</code>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10, 40</code> (retry the test every 10 seconds, but time out the test after 40 seconds). |

Comments This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

Example This example captures the text of the first Web page table and compares the data to the recorded baseline in verification point `WebTable1`. At playback, the comparison is retried every 2 seconds and times out after 30 seconds.

```
Browser SetFrame, "Type=HTMFrame;Index=2", ""
Browser NewPage, "HTMLTitle=My Web Page", ""
Result = HTMLTableVP (CompareData, "Type=HTMLTable;HTMLIndex=1",
"VP=WebTable1";Wait=2,30")
```

See Also [HTMLLink](#)

If...Then...Else

Statement

Description Executes alternative blocks of program code based on one or more expressions.

Syntax Syntax A **If** *condition* **Then** *then_statement* [**Else** *else_statement*]

Syntax B **If** *condition* **Then** *statement_block*
 [**ElseIf** *expression* **Then** *statement_block*]...
 [**Else** *statement_block*]
End If

| Syntax Element | Description |
|------------------------|--|
| <i>condition</i> | Any expression that evaluates to TRUE (non-zero) or FALSE (zero). |
| <i>then_statement</i> | Any valid single expression. |
| <i>else_statement</i> | Any valid single expression. |
| <i>expression</i> | Any expression that evaluates to TRUE (non-zero) or FALSE (zero). |
| <i>statement_block</i> | 0 or more valid expressions, separated by colons (:), or on different lines. |

Comments When multiple statements are required in either the `Then` or `Else` clauses, use the block version (Syntax B) of the `If` statement.

'\$Include

Example This example checks the time and the day of the week, and returns an appropriate message.

```
Sub main
  Dim h, m, m2, w
  h = hour(now)
  If h > 18 then
    m= "Good evening, "
  ElseIf h >12 then
    m= "Good afternoon, "
  Else
    m= "Good morning, "
  End If
  w = weekday(now)
  If w = 1 or w = 7 then m2 = "the office is closed."
  else m2 = "please hold for company operator."
  MsgBox m & m2
End Sub
```

See Also

| | |
|------------|--------------|
| Do...Loop | On...Goto |
| For...Next | Select Case |
| Goto | While...Wend |

'\$Include Metacommand

»»SQAS»

Description Includes statements from the specified header file.

Syntax **'\$Include:** *filename*

| Syntax Element | Description |
|-----------------|---|
| <i>filename</i> | The name and location of the file to include. |

Comments It is recommended (although not required) that you specify a file extension of .SBH if *filename* is a header file.

SQABasic header files can be accessed by modules within the project or within the entire repository. For information, see *Using SQABasic Header Files* in Chapter 4.

All metacommands must begin with an apostrophe (') and are recognized by the compiler only if the command starts at the beginning of a line.

Typically, the '\$Include metacommand is located before the beginning of the sub procedure. It is also possible to place the metacommand inside the sub procedure. However, it is not recommended that you do so, since compiler errors occur if the metacommand is located after a reference to a variable or constant that the included file defines, or if the included file contains a function definition used in the sub procedure.

'\$Include

If no directory or drive is specified, the compiler will search for *filename* on the source file search path.

For compatibility with other versions of Basic, you can enclose the *filename* in single quotation marks (').

A comment after an '\$Include statement results in a compiler error if the included file is enclosed in double quotes. However, a comment can be added successfully if the included file is enclosed in single quotes. For example, the first line below is correct, but the second line results in an error:

```
'$Include 'header1.sbh'           ' Compiles correctly
'$Include "header2.sbh"         ' Results in compiler error
```

Use of the colon (:) after the metacommand name is optional.

Example

This example includes a file containing the list of global variables, called GLOBALS.SBH. For this example to work correctly, you must create the GLOBALS.SBH file with at least the following statement: `Dim msgtext as String`. The `Option Explicit` statement is included in this example to prevent SQABasic from automatically dimensioning the variable as a `Variant`.

```
Option Explicit
'$Include: "c:\globals.sbh"
Sub main
    Dim msgtext as String
    gtext=InputBox("Enter a string for the global variable:")
    msgtext="The variable for the string '"
    msgtext=msgtext & gtext & "' was DIM'ed in GLOBALS.SBH."
    MsgBox msgtext
End Sub
```

See Also

```
'$CStrings
'$NoCStrings
```

InitPlay

Flow Control Command

»»SQA»

This command is obsolete in the current version of SQABasic and should no longer be used. To maintain the upward compatibility of your existing scripts, the command does not cause an error, but it has no effect on script execution.

Input (Function)

Input Function

Description Returns a string containing the characters read from a file.

Syntax **Input** [*\$*] (*number%*, [#]*filenumber%*)

| Syntax Element | Description |
|--------------------|--|
| <i>\$</i> | Optional. If specified the return type is <code>String</code> . If omitted the function will return a Variant of <code>VarType 8 (String)</code> . |
| <i>number%</i> | The number of characters to be read from the file. |
| <i>filenumber%</i> | An integer expression identifying the open file to use. |

Comments The file pointer is advanced the number of characters read. Unlike the `Input` statement, `Input` returns all characters it reads, including carriage returns, line feeds, and leading spaces.

To return a given number of bytes from a file, use `InputB`.

Example This example opens a file and prints its contents to the screen.

```
Sub main
  Dim fname
  Dim fchar()
  Dim x as Integer
  Dim msgtext
  Dim newline
  newline=Chr(10)
  On Error Resume Next
  fname=InputBox("Enter a filename to print:")
  If fname="" then
    Exit Sub
  End If
  Open fname for Input as #1
  If Err<>0 then
    MsgBox "Error loading file. Re-run program."
    Exit Sub
  End If
  msgtext="The contents of " & fname & " is: " & newline & newline
  Redim fchar(Lof(1))
  For x=1 to Lof(1)
    fchar(x)=Input(1,#1)
    msgtext=msgtext & fchar(x)
  Next x
  MsgBox msgtext
  Close #1
End Sub
```

See Also [Get](#) [Open](#)
[Input statement](#) [Write](#)
[Line Input](#)

Input Statement

Description Reads data from a sequential file and assigns the data to variables.

Syntax Syntax A **Input** [#]*filename%*, *variable*[, *variable*]...

Syntax B **Input** [*prompt\$*,] *variable*[, *variable*]...

| Syntax Element | Description |
|------------------|---|
| <i>filename%</i> | An integer expression identifying the open file to read from. |
| <i>variable</i> | The variable(s) to contain the value(s) read from the file. |
| <i>prompt\$</i> | An optional string that prompts for keyboard input. |

Comments The *filename%* is the number used in the `Open` statement to open the file. The list of *variables* is separated by commas.

If *filename%* is not specified, the user is prompted for keyboard input, either with *prompt\$* or with a question mark (?), if *prompt\$* is omitted.

Example This example prompts a user for an account number, opens a file, searches for the account number and displays the matching letter for that number. It uses the `Input` statement to increase the value of `x` and at the same time get the letter associated with each value. The second sub procedure, `CREATEFILE`, creates the file `C:\TEMP001` used by the main sub procedure.

```

Declare Sub createfile()
Global x as Integer
Global y(100) as String

Sub main
  Dim acctno as Integer
  Dim msgtext
  Call createfile
  i: acctno=InputBox("Enter an account number from 1-10:")
  If acctno<1 Or acctno>10 then
    MsgBox "Invalid account number. Try again."
    Goto i:
  End if
  x=1
  Open "C:\TEMP001" for Input as #1
  Do Until x=acctno
    Input #1, x,y(x)
  Loop
  msgtext="The letter for account number " & x & " is: " & y(x)
  Close #1
  MsgBox msgtext
  Kill "C:\TEMP001"
End Sub

```

InputBox

```
Sub createfile()  
' Put the numbers 1-10 and letters A-J into a file  
  Dim startletter  
  Open "C:\TEMP001" for Output as #1  
  startletter=65  
  For x=1 to 10  
    y(x)=Chr(startletter)  
    startletter=startletter+1  
  Next x  
  For x=1 to 10  
    Write #1, x,y(x)  
  Next x  
  Close #1  
End Sub
```

See Also Get Open
 Input function Write
 Line Input

InputBox Function

Description Displays a dialog box containing a prompt and returns a string entered by the user.

Syntax **InputBox**[\$](*prompt*\$, [*title*\$], [*default*\$], [*xpos*%, *ypos*%])

| Syntax Element | Description |
|------------------------------|--|
| \$ | Optional. If specified the return type is String. If omitted the function will return a Variant of VarType 8 (String). |
| <i>prompt</i> \$ | A string expression containing the text to show in the dialog box. |
| <i>title</i> \$ | The caption to display in the dialog box's title bar. |
| <i>default</i> \$ | The string expression to display in the edit box as the default response. |
| <i>xpos</i> %, <i>ypos</i> % | Numeric expressions, specified in dialog box units, that determine the position of the dialog box. |

Comments The length of *prompt*\$ is restricted to 255 characters. This figure is approximate and depends on the width of the characters used. Note that a carriage return and a line-feed character must be included in *prompt*\$ if a multiple-line prompt is used. If either *prompt*\$ or *default*\$ is omitted, nothing is displayed.

Xpos% determines the horizontal distance between the left edge of the screen and the left border of the dialog box. *Ypos%* determines the horizontal distance from the top of the screen to the dialog box's upper edge. If these arguments are not entered, the dialog box is centered roughly one third of the way down the screen. A horizontal dialog box unit is 1/4 of the average character width in the system font; a vertical dialog box unit is 1/8 of the height of a character in the system font.

Note: If you want to specify the dialog box's position, you must enter both of these arguments. If you enter one without the other, the default positioning is set.

If the user presses Enter, or selects the OK button, `InputBox` returns the text contained in the input box. If the user selects Cancel, the `InputBox` function returns a null string (" ").

Example This example uses `InputBox` to prompt for a file name and then prints the file name using `MsgBox`.

```
Sub main
  Dim filename
  Dim msgtext
  msgtext="Enter a filename:"
  filename=InputBox$(msgtext)
  MsgBox "The file name you entered is: " & filename
End Sub
```

See Also

| | |
|------------------------------|-------------------------------|
| Dialog Boxes | <code>MsgBox</code> function |
| <code>Input</code> function | <code>MsgBox</code> statement |
| <code>Input</code> statement | <code>PasswordBox</code> |

InputChars

User Action Command

»»SQA»

Description Sends one or more characters to the active window as if they had been entered at the keyboard.

Syntax `InputChars` *Keytext*\$

| Syntax Element | Description |
|-------------------|---|
| <i>Keytext</i> \$ | String of characters to be sent to the active window. |

InputKeys

- Comments** Do not confuse `InputChars` with `InputKeys`:
- ▶ `InputChars` treats all characters as literal characters to be entered into the active window.
 - ▶ `InputKeys` treats some characters as being representative of a keypress.
For example, if `Keytext` is `{NumDelete}`, `InputKeys` causes the Delete key on the numeric keypad to be pressed, but `InputChars` prints the literal string `{NumDelete}`.
- Strings that represent special characters (for example, `Tab` or `Enter` in Basic) can be included in an `InputChars` statement.
- Example** This example enters the characters `This is Robot.{Enter}` into the current window. Compare this example with the example for `InputKeys`.
- ```
InputChars "This is Robot.{Enter}"
```
- See Also** `InputKeys`

## InputKeys

User Action Command

»»SQAS»

- Description** Sends one or more keystrokes to the active window as if they had been entered at the keyboard.
- Syntax** `InputKeys Keytext$`
- | Syntax Element         | Description                                                                 |
|------------------------|-----------------------------------------------------------------------------|
| <code>Keytext\$</code> | String of characters representing the keys to be sent to the active window. |
- Comments** Multiple keys can be included in one `InputKeys` command — for example:
- ```
InputKeys "notepad{ENTER}"
```
- Do not confuse `InputChars` with `InputKeys`:
- ▶ `InputChars` treats all characters as literal characters to be entered into the active window.
 - ▶ `InputKeys` treats some characters as being representative of a keypress.
For example, if `Keytext` is `{NumDelete}`, `InputKeys` causes the Delete key on the numeric keypad to be pressed, but `InputChars` prints the literal string `{NumDelete}`.

Each keyboard key is represented by one or more characters. Use the following table to determine the *Keytext* value for the keyboard key you want:

| Keytext value | Keyboard equivalent |
|---|---|
| Actual printable character. Examples: A1.& | Letters A-Z, a-z, numbers 0-9, punctuation, special characters. |
| {Alt} | Default Alt key (either left or right). Default is left if there are no preceding {LKeys} or {RKeys}. |
| {Apps} | Applications key (Microsoft Natural Keyboard). |
| {LeftAlt} | Left Alt. |
| {RightAlt} | Right Alt. |
| {Backspace} <i>or</i> {BS} <i>or</i> {BkSp} | Backspace. |
| {Break} | Break or Pause. |
| {CapsLock} | Caps Lock. |
| {Clear} | Clear (key 5 on the numeric keypad when Num Lock is unlocked). |
| {Ctrl} | Default Control key (either left or right). Default is left if there are no preceding {LKeys} or {RKeys}. |
| {LeftCtrl} | Left Control. |
| {RightCtrl} | Right Control. |
| {Delete} <i>or</i> {Del} <i>or</i> {NumDelete} <i>or</i> {ExtDelete} | Delete. |
| {Down} <i>or</i> {NumDown} <i>or</i> {ExtDown} | Down Arrow. |
| {End} <i>or</i> {NumEnd} <i>or</i> {ExtEnd} | End. |
| {Enter} <i>or</i> ~ <i>or</i> {NumEnter} <i>or</i> {Num~} | Enter. |
| {Escape} <i>or</i> {Esc} | Escape. |
| {Help} | Help (a non-standard key on some PC keyboards). |

► ► ►

InputKeys

► ► ►

| Keytext value | Keyboard equivalent |
|--|---|
| {Home} <i>or</i> {NumHome} <i>or</i> {ExtHome} | Home. |
| {Insert} <i>or</i> {NumInsert} <i>or</i> {ExtInsert} | Insert. |
| {Left} <i>or</i> {NumLeft} <i>or</i> {ExtLeft} | Left Arrow. |
| {LKeys} | Sets the default for {Alt}, {Ctrl}, {Shift}, and {Win} entries as left Alt, Control, Shift, and Windows keys. |
| {Numlock} | Num Lock. |
| {PgDn} <i>or</i> {NumPgDn} <i>or</i> {ExtPgDn} | Page Down. |
| {PgUp} <i>or</i> {NumPgUp} <i>or</i> {ExtPgUp} | Page Up. |
| {PrtSc} | Print Screen. |
| {RKeys} | Sets the default for {Alt}, {Ctrl}, {Shift}, and {Win} entries as right Alt, Control, Shift, and Windows keys. |
| {Right} <i>or</i> {NumRight} <i>or</i> {ExtRight} | Right Arrow. |
| {ScrollLock} | Scroll Lock. |
| {Shift} | Default Shift key (either left or right). Default is left if there are no preceding {LKeys} or {RKeys}. |
| {LeftShift} | Left Shift. |
| {RightShift} | Right Shift. |
| {Tab} | Tab. |
| {Up} <i>or</i> {NumUp} <i>or</i> {ExtUp} | Up Arrow. |
| {Win} | Default Windows key (either left or right). Default is left if there are no preceding {LKeys} or {RKeys}. (Used on the Microsoft Natural Keyboard.) |
| {LeftWin} | Left Windows (Microsoft Natural Keyboard). |

► ► ►

▶ ▶ ▶

| Keytext value | Keyboard equivalent |
|--|--|
| {RightWin} | Right Windows (Microsoft Natural Keyboard). |
| {Num <i>n</i> }, where <i>n</i> is a number from 0 through 9 Example: {Num5} | 0-9 (numeric keypad). |
| {Num.} or . | . (period, decimal). |
| {Num-} or - | - (dash, subtraction sign). |
| {Num*} or * | * (asterisk, multiplication sign). |
| {Num/} or / | / (slash, division sign). |
| {Num+} or {+} | + (addition sign). |
| {^} | ^ (caret). |
| {%} | % (percent). |
| {~} | ~ (tilde). |
| {(} | ((left parenthesis). |
| } |) (right parenthesis). |
| {{ } | { (left brace). |
| } | } (right brace). |
| [| [(left bracket). |
|] |] (right bracket). |
| {F#} Example: {F6} | F# (function keys 1-12). |
| + Example: +{F6} | Shift (used while pressing down another key). |
| ^ Example: ^{F6} | Control (used while pressing down another key). |
| % Example: %{F6} | Alt (used while pressing down another key). |
| {key <i>n</i> }, where <i>key</i> is any key, and <i>n</i> is the number of times that <i>key</i> is pressed. Example: {a 10} | Repeats the <i>key</i> press <i>n</i> number of times. |
| {key KeyDn}, where <i>key</i> is any key. Example: {a KeyDn} | Presses and holds down <i>key</i> , and generates continuous WM_KEYDOWN events, until {key KeyUp} appears in InputKeys, or until the end of the InputKeys statement. |
| {key KeyUp}, where <i>key</i> is any key. Example: {a KeyUp} | Generates a WM_KEYUP event for <i>key</i> . |

InputKeys

Table notes:

- ▶ *Keytext* values for special words that represent keys are not case-sensitive. For example, {alt}, {Alt}, and {ALT} are all valid *Keytext* values.
- ▶ *Keytext* values with the prefix Num represent keys in the numeric keypad. *Keytext* values with the prefix Ext represent keys in the extended keypad (in between the main keypad and the numeric keypad).
- ▶ *Keytext* values for keys that appear in both the numeric and extended keypads, but do not have a Num or Ext prefix, are assumed to be in the numeric keypad.
- ▶ *Keytext* values for keys that appear in both the main and numeric keypads, but do not have a Num prefix, are assumed to be in the main keypad.
- ▶ If {CapsLock} appears in an InputKeys statement an odd number of times, the CapsLock state of the keyboard changes when the execution of InputKeys is complete. However, a {CapsLock} entry has no effect on subsequent keys within an InputKeys statement. Within InputKeys, the CapsLock state is always off.
- ▶ {Alt}, {Ctrl}, and {Shift} and the left/right designations of these keys can't be recorded. They can only be scripted manually. However, if you press Alt, Ctrl, and/or Shift in combination with other keys, Robot does record them, but as %, ^, and +, respectively.
- ▶ When {NumLock} or {ScrollLock} appears in an InputKeys statement an odd number of times, the corresponding state of the keyboard changes. However, these entries have no effect on the way subsequent keys within an InputKeys statement are recognized.
- ▶ {key KeyDn} and {key KeyUp} do not have to reflect the expected sequence of events for an actual keypress. For example, you can use {key KeyUp} with no preceding {key KeyDn}, or you can use two consecutive {key KeyUp} entries.

Example

This example enters `This is Robot.` into the current window and adds a carriage return after it. Compare this example with the example for `InputChars`.

```
InputKeys "This is Robot.{Enter}"
```

See Also

`InputChars`
`SQAQueryKey`

InStr

Function

Description Returns the position of the first occurrence of one string within another string.

Syntax Syntax A **InStr**([*start%*,] *string1*\$, *string2*\$)

 Syntax B **InStr**(*start*, *string1*\$, *string2*\$[, *compare*])

| Syntax Element | Description |
|-------------------|---|
| <i>start%</i> | The position in <i>string1</i> \$ to begin the search. (1=first character in string.) |
| <i>string1</i> \$ | The string to search. |
| <i>string2</i> \$ | The string to find. |
| <i>compare</i> | An integer expression for the method to use to compare the strings. (0=case-sensitive, 1=case-insensitive.) |

Comments If not specified, the search starts at the beginning of the string (equivalent to a *start%* of 1). *string1*\$ and *string2*\$ can be of any type. They will be converted to strings.

InStr returns a zero under the following conditions:

- ▶ *start%* is greater than the length of *string2*\$.
- ▶ *string1*\$ is a null string.
- ▶ *string2*\$ is not found.

If either *string1*\$ or *string2*\$ is a null Variant, InStr returns a null Variant.

If *string2*\$ is a null string (" "), InStr returns the value of *start%*.

If *compare* is 0, a case-sensitive comparison based on the ANSI character set sequence is performed. If *compare* is 1, a case-insensitive comparison is done based upon the relative order of characters as determined by the country code setting for your system. If *compare* is omitted, the module level default, as specified with `Option Compare`, is used.

To obtain the byte position of the first occurrence of one string within another string, use InStrB.

Int

Example This example generates a random string of characters then uses `InStr` to find the position of a single character within that string.

```
Sub main

    Dim x as Integer
    Dim y
    Dim str1 as String
    Dim str2 as String
    Dim letter as String
    Dim randomvalue
    Dim upper, lower
    Dim position as Integer
    Dim msgtext, newline

    upper=Asc("z")
    lower=Asc("a")
    newline=Chr(10)

    For x=1 to 26
        Randomize timer() + x*255
        randomvalue=Int(((upper - (lower+1)) * Rnd) +lower)
        letter=Chr(randomvalue)
        str1=str1 & letter

'Need to waste time here for fast processors
        For y=1 to 1000
            Next y

    Next x

    str2=InputBox("Enter a letter to find")
    position=InStr(str1,str2)

    If position then
        msgtext="The position of " & str2 & " is: " & position
        msgtext=msgtext & newline & "in string: " & str1
    Else
        msgtext="Letter: " & str2 & " was not found in: " & newline
        msgtext=msgtext & str1
    End If

    MsgBox msgtext

End Sub
```

See Also

| | | |
|---------------------------|-----------------------------|----------------------|
| <code>GetField</code> | <code>Mid</code> statement | <code>Str</code> |
| <code>Left</code> | <code>Option Compare</code> | <code>StrComp</code> |
| <code>Mid</code> function | <code>Right</code> | |

Int Function

Description Returns the integer part of a *number*.

Syntax **Int** (*number*)

| Syntax Element | Description |
|----------------|-------------------------|
| <i>number</i> | Any numeric expression. |

Comments For positive *numbers*, Int removes the fractional part of the expression and returns the integer part only. For negative *numbers*, Int returns the largest integer less than or equal to the expression. For example, Int (6 . 2) returns 6; Int (-6 . 2) returns -7.

The return type matches the type of the numeric expression. This includes Variant expressions that will return a result of the same VarType as input except VarType 8 (string) will be returned as VarType 5 (double) and VarType 0 (empty) will be returned as VarType 3 (long).

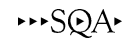
Example This example uses Int to generate random numbers in the range between the ASCII values for lowercase a and z (97 and 122). The values are converted to letters and displayed as a string.

```
Sub main
  Dim x as Integer
  Dim y
  Dim str1 as String
  Dim letter as String
  Dim randomvalue
  Dim upper, lower
  Dim msgtext, newline
  upper=Asc("z")
  lower=Asc("a")
  newline=Chr(10)
  For x=1 to 26
    Randomize timer() + x*255
    randomvalue=Int(((upper - (lower+1)) * Rnd) +lower)
    letter=Chr(randomvalue)
    str1=str1 & letter
  'Need to waste time here for fast processors
  For y=1 to 1500
    Next y
  Next x
  msgtext="The string is:" & newline
  msgtext=msgtext & str1
  MsgBox msgtext
End Sub
```

See Also Exp Rnd
Fix Sgn
Log Sqr

IPAddress

User Action Command



Description Performs an action on an IP Address control.

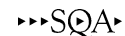
Syntax `IPAddress action%, recMethod$, parameters$`

| Syntax Element | Description |
|---------------------------|--|
| <code>action%</code> | <p>One of these mouse actions:</p> <ul style="list-style-type: none"> ▶ <i>MouseClicked</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <code>action%</code> contains a mouse-click value, <code>parameters\$</code> must contain <code>Coords=x,y</code>. ▶ <i>MouseDown</i>. The dragging of the mouse while mouse buttons and/or shifting keys (Ctrl, Alt, Shift) are pressed. When <code>action%</code> contains a mouse-drag value, <code>parameters\$</code> must contain <code>Coords=x1,y1,x2,y2</code>. <p>See Appendix E for a list of mouse click and drag values.</p> |
| <code>recMethod\$</code> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <code>ID=%</code>. The object's internal Windows ID. ▶ <code>Label=\$</code>. The text of the label object that immediately precedes the control in the internal order (Z order) of windows. ▶ <code>Name=\$</code>. A unique name that a developer assigns to an object to identify the object in the development environment. For example, the object name for a command button might be <code>Command1</code>. ▶ <code>ObjectIndex=%</code>. The number of the object among all objects of the same type in the same window. ▶ <code>Text=\$</code>. The text displayed on the object. ▶ <code>VisualText=\$</code>. An optional setting used to identify an object by its prior label. It is for user clarification only and does not affect object recognition. |
| <code>parameters\$</code> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <code>Coords=x,y</code>. If <code>action%</code> is a mouse click, specifies the coordinates of the click, relative to the top left of the object. ▶ <code>Coords=x1,y1,x2,y2</code>. If <code>action%</code> is a mouse drag, specifies the coordinates, where <code>x1,y1</code> are the starting coordinates of the drag, and <code>x2,y2</code> are the ending coordinates. The coordinates are relative to the top left of the object. |

| | |
|----------|---|
| Comments | None. |
| Example | This example clicks the IP Address control labeled "IP Address" at <i>x,y</i> coordinates of 47,5. <pre>IPAddress Click, "Label=IP Address:", "Coords=47,5"</pre> |
| See Also | IPAddressVP |

IPAddressVP

Verification Point Command



Description Establishes a verification point for an IP Address control.

Syntax *Result* = **IPAddressVP** (*action%*, *recMethod\$*, *parameters\$*)

| Syntax Element | Description |
|--------------------|--|
| <i>action%</i> | <p>The type of verification to perform. Valid values:</p> <ul style="list-style-type: none"> ▶ CompareProperties. Captures object properties information for the object and compares it to a recorded baseline. <i>parameters\$</i> VP is required; ExpectedResult and Wait are optional. |
| <i>recMethod\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ ID=%. The object's internal Windows ID. ▶ Label=\$. The text of the label object that immediately precedes the control in the internal order (Z order) of windows. ▶ Name=\$. A unique name that a developer assigns to an object to identify the object in the development environment. For example, the object name for a command button might be Command1. ▶ ObjectIndex=%. The number of the object among all objects of the same type in the same window. ▶ Text=\$. The text displayed on the object. ▶ VisualText=\$. An optional setting used to identify an object by its prior label. It is for user clarification only and does not affect object recognition. |

▶ ▶ ▶

IPmt

► ► ►

| Syntax Element | Description |
|----------------------|---|
| <i>parameters</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> ► ExpectedResult=%. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> – PASS. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail. – FAIL. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail. ► VP=\$. The verification point ID. IDs must be unique within a script. Required for all verification points. ► Wait=%,%. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10,40</code> (retry the test every 10 seconds, but time out the test after 40 seconds). |

Comments This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

Example This example captures the properties of the IP Address calendar control labeled "IP Address" and compares them to the recorded baseline in verification point IPADDR1.

```
Result = IPAddressVP (CompareProperties, "Label=IP Address:",
"VP=IPADDR1")
```

See Also IPAddress

IPmt Function

Description Returns the interest portion of a payment for a given period of an annuity.

Syntax **IPmt**(*rate*, *per*, *nper*, *pv*, *fv*, *due*)

| Syntax Element | Description |
|----------------|--|
| <i>rate</i> | Interest rate per period. |
| <i>per</i> | Particular payment period in the range 1 through <i>nper</i> . |

► ► ►

▶ ▶ ▶

| Syntax Element | Description |
|----------------|---|
| <i>nper</i> | Total number of payment periods. |
| <i>pv</i> | Present value of the initial lump sum amount paid (as in the case of an annuity) or received (as in the case of a loan). |
| <i>fv</i> | Future value of the final lump sum amount required (as in the case of a savings plan) or paid (0 as in the case of a loan). |
| <i>due</i> | 0 if payments are due at the end of each payment period, and 1 if they are due at the beginning of the period. |

Comments The given interest rate is assumed constant over the life of the annuity. If payments are on a monthly schedule, then *rate* will be 0.0075 if the annual percentage rate on the annuity or loan is 9%.

Example This example finds the interest portion of a loan payment amount for payments made in last month of the first year. The loan is for \$25,000 to be paid back over 5 years at 9.5% interest.

```

Sub main
  Dim aprate, periods
  Dim payperiod
  Dim loanpv, due
  Dim loanfv, intpaid
  Dim msgtext
  aprate=.095
  payperiod=12
  periods=120
  loanpv=25000
  loanfv=0
  Rem Assume payments are made at end of month
  due=0
  intpaid=IPmt(aprate/12,payperiod,periods,-loanpv,loanfv,due)
  msgtext="For a loan of $25,000 @ 9.5% for 10 years," & Chr(10)
  msgtext=msgtext+ "the interest paid in month 12 is: "
  msgtext=msgtext + Format(intpaid, "Currency")
  MsgBox msgtext
End Sub

```

See Also

| | |
|-----|------|
| FV | Pmt |
| IRR | PV |
| NPV | Rate |
| Pmt | |

IRR

IRR

Function

Description Returns the internal rate of return for a stream of periodic cash flows.

Syntax `IRR(valuearray(), guess)`

| Syntax Element | Description |
|---------------------------|---|
| <code>valuearray()</code> | An array containing cash flow values. |
| <code>guess</code> | A ballpark estimate of the value returned by IRR. |

Comments `valuearray()` must have at least one positive value (representing a receipt) and one negative value (representing a payment). All payments and receipts must be represented in the exact sequence. The value returned by IRR will vary with the change in the sequence of cash flows.

In general, a `guess` value of between 0.1 (10 percent) and 0.15 (15 percent) would be a reasonable estimate.

IRR is an iterative function. It improves a given guess over several iterations until the result is within 0.00001 percent. If it does not converge to a result within 20 iterations, it signals failure.

Example This example calculates an internal rate of return (expressed as an interest rate percentage) for a series of business transactions (income and costs). The first value entered must be a negative amount, or IRR generates an `Illegal Function Call error`.

```
Sub main
  Dim cashflows() as Double
  Dim guess, count as Integer
  Dim i as Integer
  Dim intnl as Single
  Dim msgtext as String
  guess=.15
  count=InputBox("How many cash flow amounts do you have?")
  ReDim cashflows(count+1)
  For i=0 to count-1
    cashflows(i)=InputBox("Enter income for month " & i+1 & ":")
  Next i
  intnl=IRR(cashflows(),guess)
  msgtext="The IRR for your cash flow amounts is: "
  msgtext=msgtext & Format(intnl, "Percent")
  MsgBox msgtext
End Sub
```

See Also

| | |
|------|------|
| FV | PPmt |
| IPmt | PV |
| NPV | Rate |
| Pmt | |

Is Operator

Description Compares two object expressions and returns -1 if they refer to the same object, 0 otherwise.

Syntax *objectExpression* **Is** *objectExpression*

| Syntax Element | Description |
|-------------------------|------------------------------|
| <i>objectExpression</i> | Any valid object expression. |

Comments **Is** can also be used to test if an object variable has been Set to Nothing.

Example This example displays a list of open files in the software application, VISIO. It uses the **Is** operator to determine whether VISIO is available. To see how this example works, you need to start VISIO and open one or more documents.

```
Sub main
    Dim visio as Object
    Dim doc as Object
    Dim msgtext as String
    Dim i as Integer, doccount as Integer

    'Initialize Visio
    Set visio = GetObject("visio.application") ' find Visio
    If (visio Is Nothing) then
        MsgBox "Couldn't find Visio!"
        Exit Sub
    End If
    'Get # of open Visio files
    doccount = visio.documents.count           'OLE2 call to Visio
    If doccount=0 then
        msgtext="No open Visio documents."
    Else
        msgtext="The open files are: " & Chr$(13)
        For i = 1 to doccount
            ' access Visio's doc method
            Set doc = visio.documents(i)
            msgtext=msgtext & Chr$(13) & doc.name
        Next i
    End If
    MsgBox msgtext
End Sub
```

See Also

| | |
|---------------|---------|
| Class List | Nothing |
| Create Object | Object |
| Get Object | Typeof |

IsDate

IsDate

Function

Description Returns -1 (TRUE) if an expression is a legal date, 0 (FALSE) if it is not.

Syntax **IsDate**(*expression*)

| Syntax Element | Description |
|-------------------|-----------------------|
| <i>expression</i> | Any valid expression. |

Comments **IsDate** returns -1 (TRUE) if the expression is of `VarType 7` (date) or a string that can be interpreted as a date.

Example This example accepts a string from the user and checks to see if it is a valid date.

```
Sub main
  Dim theDate
  theDate = InputBox("Enter a date:")
  If IsDate(theDate) = -1 Then
    MsgBox "The new date is: " & Format(CVDate(theDate), "dddddd")
  Else
    MsgBox "The date is not valid."
  End If
End Sub
```

See Also `CVDate` `IsNumeric`
`IsEmpty` `VarType`
`IsNull`

IsEmpty

Function

Description Returns -1 (TRUE) if a Variant has been initialized. 0 (FALSE) otherwise.

Syntax **IsEmpty**(*expression*)

| Syntax Element | Description |
|-------------------|---|
| <i>expression</i> | Any expression with a data type of Variant. |

Comments `IsEmpty` returns -1 (TRUE) if the Variant is of `VarType 0` (empty). Any newly-defined Variant defaults to being of Empty type, to signify that it contains no initialized data. An Empty Variant converts to zero when used in a numeric expression, or an empty string (" ") in a string expression.

Example This example prompts for a series of test scores and uses `IsEmpty` to determine whether the maximum allowable limit has been hit. (`IsEmpty` determines when to exit the `Do . . . Loop`.)

```
Sub main
  Dim arrayvar(10)
  Dim x as Integer
  Dim msgtext as String
  Dim tscore as Single
  Dim total as Integer
  x=1
  Do
    tscore=InputBox("Enter test score #" & x & ":")
    arrayvar(x)=tscore
    x=x+1
  Loop Until IsEmpty(arrayvar(10))<>-1
  total=x-1
  msgtext="You entered: " & Chr(10)
  For x=1 to total
    msgtext=msgtext & Chr(10) & arrayvar(x)
  Next x
  MsgBox msgtext
End Sub
```

See Also `IsDate` `IsNumeric`
`IsNull` `VarType`

IsMissing

Function

Description Returns -1 (TRUE) if an optional argument was not supplied by the user, 0 (FALSE) otherwise.

Syntax `IsMissing`(*argname*)

| Syntax Element | Description |
|----------------|---|
| <i>argname</i> | An optional argument for an SQABasic command. |

IsNull

Comments `IsMissing` is used in procedures that have optional arguments to find out whether the argument's value was supplied or not.

Example This example prints a list of letters. The number printed is determined by the user. If the user wants to print all letters, the sub procedure `myfunc` is called without any argument. The sub procedure uses `IsMissing` to determine whether to print all the letters or just the number specified by the user.

```
Sub myfunc(Optional arg1)
  If IsMissing(arg1)=-1 then
    arg1=26
  End If
  msgtext="The letters are: " & Chr$(10)
  For x= 1 to arg1
    msgtext=msgtext & Chr$(x+64) & Chr$(10)
  Next x
  MsgBox msgtext
End Sub

Sub Main
  Dim arg1
  arg1=InputBox("How many letters to print (0 for all):")
  If arg1=0 then
    myfunc
  Else
    myfunc arg1
  End If
End Sub
```

See Also `Function...End Function`

IsNull

Function

Description Returns -1 (TRUE) if a Variant expression contains the `Null` value, 0 (FALSE) otherwise.

Syntax `IsNull(expression)`

| Syntax Element | Description |
|-------------------|---|
| <i>expression</i> | Any expression with a data type of Variant. |

Comments Null Variants have no associated data and serve only to represent invalid or ambiguous results. `Null` is not the same as `Empty`, which indicates that a Variant has not yet been initialized.

Example This example asks for ten test score values and calculates the average. If any score is negative, the value is set to `Null`. Then `IsNull` is used to reduce the total count of scores (originally 10) to just those with positive values before calculating the average.

```

Sub main
  Dim arrayvar(10)
  Dim count as Integer
  Dim total as Integer
  Dim x as Integer
  Dim msgtext as String
  Dim tscore as Single
  count=10
  total=0
  For x=1 to count
    tscore=InputBox("Enter test score #" & x & ":")
    If tscore<0 then
      arrayvar(x)=Null
    Else
      arrayvar(x)=tscore
      total=total+arrayvar(x)
    End If
  Next x
  Do While x<>0
    x=x-1
    If IsNull(arrayvar(x))=-1 then
      count=count-1
    End If
  Loop
  msgtext="The average (excluding negative values) is: " & Chr(10)
  msgtext=msgtext & Format (total/count, "##.##")
  MsgBox msgtext
End Sub

```

See Also `IsDate` `IsNumeric`
`IsEmpty` `VarType`

IsNumeric

Function

Description Returns -1 (TRUE) if an expression has a data type of Numeric, 0 (FALSE) otherwise.

Syntax `IsNumeric(expression)`

| Syntax Element | Description |
|-------------------|-----------------------|
| <i>expression</i> | Any valid expression. |

JavaCanvas

Comments `IsNumeric` returns -1 (TRUE) if the expression is of `VarType 2` through `VarType 6` (numeric) or a string that can be interpreted as a number.

Example This example uses `IsNumeric` to determine whether a user selected an option (1-3) or typed Q to quit.

```
Sub main
  Dim answer
  answer=InputBox("Enter a choice (1-3) or type Q to quit")
  If IsNumeric(answer)=-1 then
    Select Case answer
      Case 1
        MsgBox "You chose #1."
      Case 2
        MsgBox "You chose #2."
      Case 3
        MsgBox "You chose #3."
    End Select
  Else
    MsgBox "You typed Q."
  End If
End Sub
```

See Also `IsDate` `IsNull`
 `IsEmpty` `VarType`

JavaCanvas

User Action Command

»»SQAS»

Description Performs an action on a Java canvas component.

Syntax **JavaCanvas** *action%*, *recMethod\$*, *parameters\$*

| Syntax Element | Description |
|----------------|--|
| <i>action%</i> | One of these mouse actions: <ul style="list-style-type: none">▶ <i>MouseClicked</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> must contain <code>Coords=x,y,AreaIndex=%,</code> or <code>AreaName=\$</code>.▶ <i>MouseDown</i>. The dragging of the mouse while mouse buttons and/or shifting keys (Ctrl, Alt, Shift) are pressed. When <i>action%</i> contains a mouse-drag value, <i>parameters\$</i> must contain <code>Coords=x1,y1,x2,y2</code>. See Appendix E for a list of mouse click and drag values. |

▶ ▶ ▶



| Syntax Element | Description |
|---------------------------|---|
| <code>recMethod\$</code> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <code>Index=%</code>. The number of the parent or child object among all objects identified with the same base recognition method. Typically, <code>Index</code> is used after another recognition method qualifier — for example, <code>Name=\$; Index=%</code>. ▶ <code>JavaText=\$</code>. A label that identifies the child object in the user interface. ▶ <code>Name=\$</code>. A name that a developer assigns to a parent or child object to identify the object. For example, the object name for a command button might be <code>Command1</code>. ▶ <code>Type=\$</code>. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The <code>Type</code> qualifier uses the following form: <code>Type=\$; recMethod=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>;\;</code>). <p>See <i>Recognition Methods in Java Commands</i> in Chapter 4 for other recognition methods that specify the parent object.</p> |
| <code>parameters\$</code> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <code>AreaIndex=%</code>. An ID assigned to a Java canvas. The number of the area among all areas of the same type within a Java canvas. Used with client-side canvasses. ▶ <code>AreaName=\$</code>. A name assigned to an area in a Java canvas. Used with client-side canvasses. ▶ <code>Coords=x,y</code>. If <code>action%</code> is a mouse click, specifies the coordinates of the click, relative to the top left of the object. Robot uses this parameter only if the item contents or index cannot be retrieved — for example, if the list view is empty or disabled. ▶ <code>Coords=x1,x2,y1,y2</code>. If <code>action%</code> is a mouse drag, specifies the coordinates, where <code>x1,y1</code> are the starting coordinates of the drag, and <code>x2,y2</code> are the ending coordinates. The coordinates are relative to the top left of the object. |

Comments In earlier releases of Robot, Java canvas components were treated as Java panel components. Consequently, for backward compatibility, the recognition method value `Index=%` includes panel components as well as canvas components. For example, a canvas component that is the first canvas component but that is nested

JavaCanvasVP

inside several panels can be specified as `Index=4` — because the panel components are included in the index.

If the parent object is not specified in the `recMethod` argument of this command, it must be specified in a preceding `Browser` command. For more information about specifying parent and child Java objects, see *Recognition Methods in Java Commands* in Chapter 4.

Example This example performs a left-mouse click at the specified coordinates relative to the top left corner of the canvas component.

```
JavaCanvas Click,  
  "JavaCaption="Sample App\;\Type=JavaCanvas;Index=3",  
  "Coords=10,16"
```

See Also JavaCanvasVP

JavaCanvasVP

Verification Point Command

»»SQA»

Description Establishes a verification point for a Java canvas component.

Syntax `Result = JavaCanvasVP(action%, recMethod$, parameters$)`

| Syntax Element | Description |
|--------------------------|---|
| <code>action%</code> | The type of verification to perform. Valid values: <ul style="list-style-type: none">▶ <code>CompareData</code>. Captures the data of the object and compares it to a recorded baseline. <code>parameters\$ VP</code> is required; <code>ExpectedResult</code> and <code>Wait</code> are optional.▶ <code>CompareProperties</code>. Captures object properties information for the object and compares it to a recorded baseline. <code>parameters\$ VP</code> is required; <code>ExpectedResult</code> and <code>Wait</code> are optional. |
| <code>recMethod\$</code> | Valid values: <ul style="list-style-type: none">▶ <code>Index=%</code>. The number of the parent or child object among all objects identified with the same base recognition method. Typically, <code>Index</code> is used after another recognition method qualifier — for example, <code>Name=\$; Index=%</code>.▶ <code>JavaText=\$</code>. A label that identifies the child object in the user interface.▶ <code>Name=\$</code>. A name that a developer assigns to a parent or child object to identify the object. For example, the object name for a command button might be <code>Command1</code>. |

▶ ▶ ▶



| Syntax Element | Description |
|---------------------|---|
| | <ul style="list-style-type: none"> ▶ <code>Type=\$</code>. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The <code>Type</code> qualifier uses the following form: <code>Type=\$;recMethod=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>;\;</code>). <p>See <i>Recognition Methods in Java Commands</i> in Chapter 4 for other recognition methods that specify the parent object.</p> |
| <i>parameters\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <code>ExpectedResult=%</code>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> – <code>PASS</code>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail. – <code>FAIL</code>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail. ▶ <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points. ▶ <code>Wait=%,%</code>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10,40</code> (retry the test every 10 seconds, but time out the test after 40 seconds). |

Comments This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

If the parent object is not specified in the *recMethod* argument of this command, it must be specified in a preceding *Browser* command. For more information about specifying parent and child Java objects, see *Recognition Methods in Java Commands* in Chapter 4.

Example This example captures the properties of a Java canvas component.

```
Result = JavaCanvasVP(CompareProperties,
    "JavaCaption="Sample App;\;Type=JavaCanvas;Index=4,
    "VP=Object Properties")
```

See Also JavaCanvas

JavaListView

User Action Command

Description Performs an action on a Java multi-column list component.

Syntax **JavaListView** *action%*, *recMethod\$*, *parameters\$*

| Syntax Element | Description |
|----------------|---|
| <i>action%</i> | <p>One of these actions:</p> <ul style="list-style-type: none"> ▶ Deselect. Deselects the specified item from an extended Java list view component in <code>multipleMode</code>. <i>recMethod\$</i> must contain one of the Java recognition methods, and <i>parameters\$</i> must contain either <code>Text</code> or <code>Index</code>. ▶ ExtendSelection. Selects the specified item from an extended Java list view component in <code>multipleMode</code>. <i>recMethod\$</i> must contain one of the Java recognition methods, and <i>parameters\$</i> must contain either <code>Text</code> or <code>Index</code>. ▶ MakeSelection. Selects the specified item in a Java list view. <i>recMethod\$</i> must contain one of the Java recognition methods, and <i>parameters\$</i> must contain either <code>Text</code> or <code>Index</code>. ▶ MouseClicked. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> must contain <code>Coords=x,y</code>. ▶ MouseDown. The dragging of the mouse while mouse buttons and/or shifting keys (Ctrl, Alt, Shift) are pressed. When <i>action%</i> contains a mouse-drag value, <i>parameters\$</i> must contain <code>Coords=x1,y1,x2,y2</code>. See Appendix E for a list of mouse click and drag values. ▶ ScrollAction. One of these scroll actions: <ul style="list-style-type: none"> <code>ScrollPageRight</code> <code>ScrollPageDown</code> <code>ScrollRight</code> <code>ScrollLineDown</code> <code>ScrollPageLeft</code> <code>ScrollPageUp</code> <code>ScrollLeft</code> <code>ScrollLineUp</code> <code>HScrollTo</code> <code>VScrollTo</code> <p>HScrollTo and VScrollTo take the required parameter <code>Position=%</code>.</p> <p>If Robot cannot interpret the action being applied to a scroll bar, which happens with certain custom standalone scroll bars, it records the action as a click or drag.</p> |

▶ ▶ ▶

▶ ▶ ▶

| Syntax Element | Description |
|---------------------------|--|
| <code>recMethod\$</code> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <code>Index=%</code>. The number of the parent or child object among all objects identified with the same base recognition method. Typically, <code>Index</code> is used after another recognition method qualifier — for example, <code>Name=\$; Index=%</code>. ▶ <code>JavaText=\$</code>. A label that identifies the child object in the user interface. ▶ <code>Name=\$</code>. A name that a developer assigns to a parent or child object to identify the object. For example, the object name for a command button might be <code>Command1</code>. ▶ <code>Type=\$</code>. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The <code>Type</code> qualifier uses the following form: <code>Type=\$; recMethod=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>;\;</code>). <p>See <i>Recognition Methods in Java Commands</i> in Chapter 4 for other recognition methods that specify the parent object.</p> |
| <code>parameters\$</code> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <code>Coords=x,y</code>. If <code>action%</code> is a mouse click, specifies the coordinates of the click, relative to the top left of the object. Robot uses this parameter only if the item contents or index cannot be retrieved — for example, if the list view is empty or disabled. ▶ <code>Coords=x1,x2,y1,y2</code>. If <code>action%</code> is a mouse drag, specifies the coordinates, where <code>x1,y1</code> are the starting coordinates of the drag, and <code>x2,y2</code> are the ending coordinates. The coordinates are relative to the top left of the object. ▶ <code>Index=%</code>. If <code>action%</code> is a select or deselect action, identifies the index of an item in the list. ▶ <code>Position=%</code>. If <code>action%</code> is a <code>VScrollTo</code> or <code>HScrollTo</code>, specifies the scroll bar value of the new scrolled-to position. Every scroll bar has an internal range, and this value is specific to that range. ▶ <code>Text=\$</code>. If <code>action%</code> is a select or deselect action, identifies the text of an item in the list. |

JavaListViewVP

Comments If the parent object is not specified in the *recMethod* argument of this command, it must be specified in a preceding *Browser* command. For more information about specifying parent and child Java objects, see *Recognition Methods in Java Commands* in Chapter 4.

Example This example selects a row in a Java list view component.

```
JavaListView MakeSelection,  
  "JavaCaption=Sample App;\;Type=JavaListView;Index=1",  
  "Text=Hooked on Java"
```

See Also JavaListViewVP

JavaListViewVP

Verification Point Command

»»SQA»

Description Establishes a verification point for a Java multi-column list component.

Syntax *Result* = **JavaListViewVP**(*action%*, *recMethod\$*, *parameters\$*)

| Syntax Element | Description |
|--------------------|---|
| <i>action%</i> | The type of verification to perform. Valid values: <ul style="list-style-type: none">▶ CompareData. Captures the data of the object and compares it to a recorded baseline. <i>parameters\$ VP</i> is required; <i>ExpectedResult</i> and <i>Wait</i> are optional.▶ CompareProperties. Captures object properties information for the object and compares it to a recorded baseline. <i>parameters\$ VP</i> is required; <i>ExpectedResult</i> and <i>Wait</i> are optional. |
| <i>recMethod\$</i> | Valid values: <ul style="list-style-type: none">▶ Index=%. The number of the parent or child object among all objects identified with the same base recognition method. Typically, <i>Index</i> is used after another recognition method qualifier — for example, <i>Name=\$; Index=%</i>.▶ JavaText=\$. A label that identifies the child object in the user interface.▶ Name=\$. A name that a developer assigns to a parent or child object to identify the object. |

▶ ▶ ▶



| Syntax Element | Description |
|---------------------------|---|
| | <ul style="list-style-type: none"> ▶ <code>Type=\$</code>. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The <code>Type</code> qualifier uses the following form: <code>Type=\$;recMethod=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>;\</code>). <p>See <i>Recognition Methods in Java Commands</i> in Chapter 4 for other recognition methods that specify the parent object.</p> |
| <code>parameters\$</code> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <code>ExpectedResult=%</code>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> – <code>PASS</code>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail. – <code>FAIL</code>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail. ▶ <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points. ▶ <code>Wait=%,%</code>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10,40</code> (retry the test every 10 seconds, but time out the test after 40 seconds). |

Comments This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

If the parent object is not specified in the `recMethod` argument of this command, it must be specified in a preceding `Browser` command. For more information about specifying parent and child Java objects, see *Recognition Methods in Java Commands* in Chapter 4.

Example This example captures the properties of a Java list view component.

```
Result = JavaListViewVP(CompareProperties,
    "JavaCaption=Sample App;\;Type=JavaListView;Index=1",
    VP=ObjectProperties")
```

See Also `JavaListView`

JavaMenu

User Action Command

»»SQA»

Description Performs an action on a Java menu.

Syntax **JavaMenu** *action%*, *recMethod\$*, *parameters\$*

| Syntax Element | Description |
|--------------------|---|
| <i>action%</i> | <p>One of these mouse actions:</p> <ul style="list-style-type: none"> ▶ <i>MakeSelection</i>. Selects the specified item from a Java menu. ▶ <i>MouseClicked</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> must contain <i>Coords=x,y</i>. <p>See Appendix E for a list of mouse click values.</p> |
| <i>recMethod\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <i>Index=%</i>. The number of the parent or child object among all objects identified with the same base recognition method. Typically, <i>Index</i> is used after another recognition method qualifier — for example, <i>Name=\$;Index=%</i>. ▶ <i>JavaText=\$</i>. A label that identifies the child object in the user interface. ▶ <i>Name=\$</i>. A name that a developer assigns to a parent or child object to identify the object. For example, the object name for a command button might be <i>Command1</i>. ▶ <i>Path=\$</i>. If <i>action%</i> is <i>MakeSelection</i>, identifies the text of the item as a path. Sub-menus are separated by a pointer (<i>-></i>). ▶ <i>Type=\$</i>. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The <i>Type</i> qualifier uses the following form: <i>Type=\$;recMethod=\$</i>. Parent/child values are separated by a backslash and semicolons (<i>;\;</i>). <p>See <i>Recognition Methods in Java Commands</i> in Chapter 4 for other recognition methods that specify the parent object.</p> |

▶ ▶ ▶

▶ ▶ ▶

| Syntax Element | Description |
|---------------------|---|
| <i>parameters\$</i> | Valid value: ▶ <i>Coords=x,y</i> . If <i>action%</i> is a mouse click, specifies the coordinates of the click, relative to the top left of the object. |

Comments Robot can recognize menus and sub-menus up to five levels deep.

If the parent object is not specified in the *recMethod* argument of this command, it must be specified in a preceding *Browser* command. For more information about specifying parent and child Java objects, see *Recognition Methods in Java Commands* in Chapter 4.

Example This example selects the Java menu option Color Chooser from the Choosers menu. The menu bar is located within the Java applet named Main.

```
Window SetContext, "Caption=Java demo", ""
Browser SetApplet, "Name=Main", ""
JavaMenu MakeSelection, "Type=JavaMenu;Name=Swing
menus;Path=Choosers->Color Chooser", ""
```

See Also JavaMenuVP

JavaMenuVP

Verification Point Command

▶▶SQA▶

Description Establishes a verification point for a Java menu.

Syntax *Result* = **JavaMenuVP** (*action%*, *recMethod\$*, *parameters\$*)

| Syntax Element | Description |
|----------------|---|
| <i>action%</i> | The type of verification to perform. Valid value: ▶ <i>CompareProperties</i> . Captures object properties information for the object and compares it to a recorded baseline. <i>parameters\$ VP</i> is required; <i>ExpectedResult</i> and <i>Wait</i> are optional. |

▶ ▶ ▶



| Syntax Element | Description |
|---------------------------|---|
| <code>recMethod\$</code> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <code>Index=%</code>. The number of the parent or child object among all objects identified with the same base recognition method. Typically, <code>Index</code> is used after another recognition method qualifier — for example, <code>Name=\$; Index=%</code>. ▶ <code>JavaText=\$</code>. A label that identifies the child object in the user interface. ▶ <code>Name=\$</code>. A name that a developer assigns to a parent or child object to identify the object. For example, the object name for a command button might be <code>Command1</code>. ▶ <code>Path=\$</code>. Identifies the text of the item as a path. Submenus are separated by a pointer (<code>-></code>). ▶ <code>Type=\$</code>. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The <code>Type</code> qualifier uses the following form: <code>Type=\$; recMethod=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>;\;</code>). <p>See <i>Recognition Methods in Java Commands</i> in Chapter 4 for other recognition methods that specify the parent object.</p> |
| <code>parameters\$</code> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <code>ExpectedResult=%</code>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> – <code>PASS</code>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail. – <code>FAIL</code>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail. ▶ <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points. ▶ <code>Wait=% , %</code>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10 , 40</code> (retry the test every 10 seconds, but time out the test after 40 seconds). |

Comments This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

Robot can recognize menus and sub-menus up to five levels deep.

If the parent object is not specified in the *recMethod* argument of this command, it must be specified in a preceding *Browser* command. For more information about specifying parent and child Java objects, see *Recognition Methods in Java Commands* in Chapter 4.

Example This example captures the properties of the Java menu with a Name attribute of MainMenu. The menu is located within the Java applet named Main. JavaMenuVP compares the properties to the recorded baseline in verification point MENUVP1. At playback, the comparison is retried every 2 seconds and times out after 30 seconds.

```
Window SetContext, "Caption=Java demo", ""
Browser SetApplet, "Name=Main", ""
Result = JavaMenuVP(CompareProperties, "Type=JavaMenu;Name=MainMenu",
"VP=MENUVP1;Wait=2,30")
```

See Also JavaMenu

JavaObject

User Action Command



Description Performs an action on an unrecognized Java component.

Syntax **JavaObject** *action%*, *recMethod\$*, *parameters\$*

| Syntax Element | Description |
|----------------|--|
| <i>action%</i> | <p>One of these mouse actions:</p> <ul style="list-style-type: none"> ▶ <i>MouseClicked</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> must contain <i>Coords=x,y</i>. ▶ <i>MouseDown</i>. The dragging of the mouse while mouse buttons and/or shifting keys (Ctrl, Alt, Shift) are pressed. When <i>action%</i> contains a mouse-drag value, <i>parameters\$</i> must contain <i>Coords=x1,y1,x2,y2</i>. <p>See Appendix E for a list of mouse click and drag values.</p> <p style="text-align: right;">▶ ▶ ▶</p> |

JavaObject

▶ ▶ ▶

| Syntax Element | Description |
|---------------------------|--|
| <code>recMethod\$</code> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <code>Index=%</code>. The number of the parent or child object among all objects identified with the same base recognition method. Typically, <code>Index</code> is used after another recognition method qualifier — for example, <code>Name=\$; Index=%</code>. ▶ <code>JavaText=\$</code>. A label that identifies the child object in the user interface. ▶ <code>Name=\$</code>. A name that a developer assigns to a parent or child object to identify the object. ▶ <code>Type=\$</code>. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The <code>Type</code> qualifier uses the following form: <code>Type=\$; recMethod=\$</code>. Parent/child values are separated by a backslash and semicolons (<code> ; \ ;</code>). <p>See <i>Recognition Methods in Java Commands</i> in Chapter 4 for other recognition methods that specify the parent object.</p> |
| <code>parameters\$</code> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <code>Coords=x,y</code>. If <code>action%</code> is a mouse click, specifies the coordinates of the click, relative to the top left of the object. ▶ <code>Coords=x1,y1,x2,y2</code>. If <code>action%</code> is a mouse drag, specifies the coordinates, where <code>x1,y1</code> are the starting coordinates of the drag, and <code>x2,y2</code> are the ending coordinates. The coordinates are relative to the top left of the object. |

Comments If the parent object is not specified in the `recMethod` argument of this command, it must be specified in a preceding `Browser` command. For more information about specifying parent and child Java objects, see *Recognition Methods in Java Commands* in Chapter 4.

Example This example clicks a Java object titled `MyObject` at coordinates `20,40`. The object is located within the Java applet named `Main`.

```
Window SetContext, "Caption=Java demo", ""
Browser SetApplet, "Name=Main", ""
JavaObject Click, "Type=JavaObject ; Name=MyObject", "Coords=20,40"
```

See Also `JavaObjectVP`

JavaObjectVP

Verification Point Command



Description Establishes a verification point for an unrecognized Java component.

Syntax `Result = JavaObjectVP (action%, recMethod$, parameters$)`

| Syntax Element | Description |
|---------------------------|--|
| <code>action%</code> | <p>The type of verification to perform. Valid value:</p> <ul style="list-style-type: none"> ▶ <code>CompareProperties</code>. Captures object properties information for the object and compares it to a recorded baseline. <code>parameters\$ VP</code> is required; <code>ExpectedResult</code> and <code>Wait</code> are optional. |
| <code>recMethod\$</code> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <code>Index=%</code>. The number of the parent or child object among all objects identified with the same base recognition method. Typically, <code>Index</code> is used after another recognition method qualifier — for example, <code>Name=\$; Index=%</code>. ▶ <code>JavaText=\$</code>. A label that identifies the child object in the user interface. ▶ <code>Name=\$</code>. A name that a developer assigns to a parent or child object to identify the object. ▶ <code>Type=\$</code>. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The <code>Type</code> qualifier uses the following form: <code>Type=\$; recMethod=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>;\</code>). <p>See <i>Recognition Methods in Java Commands</i> in Chapter 4 for other recognition methods that specify the parent object.</p> |
| <code>parameters\$</code> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <code>ExpectedResult=%</code>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> – <code>PASS</code>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail. – <code>FAIL</code>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail. |



JavaPanel

▶ ▶ ▶

| Syntax Element | Description |
|----------------|--|
| | <ul style="list-style-type: none">▶ <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points.▶ <code>Wait=%,%</code>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10,40</code> (retry the test every 10 seconds, but time out the test after 40 seconds). |

Comments This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

If the parent object is not specified in the *recMethod* argument of this command, it must be specified in a preceding `Browser` command. For more information about specifying parent and child Java objects, see *Recognition Methods in Java Commands* in Chapter 4.

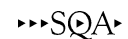
Example This example captures the properties of the Java object named MyObject. The object is located within the Java applet named Main. `JavaObjectVP` compares the properties to the recorded baseline in verification point `JOBJECTVP1`. At playback, the comparison is retried every 2 seconds and times out after 30 seconds.

```
Window SetContext, "Caption=Java demo", ""
Browser SetApplet, "Name=Main", ""
Result = JavaObjectVP (CompareProperties,
    "Type=JavaObject;Name=MyObject", "VP=JOBJECTVP1;Wait=2,30")
```

See Also `JavaObject`

JavaPanel

User Action Command



Description Performs an action on a Java panel or canvas.

Syntax `JavaPanel action%, recMethod$, parameters$`

| Syntax Element | Description |
|---------------------|--|
| <i>action%</i> | <p>One of these mouse actions:</p> <ul style="list-style-type: none"> ▶ <i>MouseClicked</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> must contain <i>Coords=x,y</i>. ▶ <i>MouseDown</i>. The dragging of the mouse while mouse buttons and/or shifting keys (Ctrl, Alt, Shift) are pressed. When <i>action%</i> contains a mouse-drag value, <i>parameters\$</i> must contain <i>Coords=x1,y1,x2,y2</i>. <p>See Appendix E for a list of mouse click and drag values.</p> |
| <i>recMethod\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <i>Index=%</i>. The number of the parent or child object among all objects identified with the same base recognition method. Typically, <i>Index</i> is used after another recognition method qualifier — for example, <i>Name=\$; Index=%</i>. ▶ <i>JavaText=\$</i>. A label that identifies the child object in the user interface. ▶ <i>Name=\$</i>. A name that a developer assigns to a parent or child object to identify the object. ▶ <i>Type=\$</i>. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The <i>Type</i> qualifier uses the following form: <i>Type=\$; recMethod=\$</i>. Parent/child values are separated by a backslash and semicolons (<i>;\</i>). <p>See <i>Recognition Methods in Java Commands</i> in Chapter 4 for other recognition methods that specify the parent object.</p> |
| <i>parameters\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <i>Coords=x,y</i>. If <i>action%</i> is a mouse click, specifies the coordinates of the click, relative to the top left of the object. ▶ <i>Coords=x1,y1,x2,y2</i>. If <i>action%</i> is a mouse drag, specifies the coordinates, where <i>x1,y1</i> are the starting coordinates of the drag, and <i>x2,y2</i> are the ending coordinates. The coordinates are relative to the top left of the object. |

Comments If the parent object is not specified in the *recMethod* argument of this command, it must be specified in a preceding *Browser* command. For more

JavaPanelVP

information about specifying parent and child Java objects, see *Recognition Methods in Java Commands* in Chapter 4.

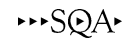
Example This example clicks the panel titled EmployeeList at coordinates 25,50. The panel is located within the Java applet named Main.

```
Window SetContext, "Caption=Java demo", ""
Browser SetApplet, "Name=Main", ""
JavaPanel Click, "Type=JavaPanel;Name=EmployeeList", "Coords=25,50"
```

See Also JavaPanelVP

JavaPanelVP

Verification Point Command



Description Establishes a verification point for a Java panel or canvas.

Syntax `Result = JavaPanelVP (action%, recMethod$, parameters$)`

| Syntax Element | Description |
|--------------------------|---|
| <code>action%</code> | The type of verification to perform. Valid value: <ul style="list-style-type: none">▶ <code>CompareProperties</code>. Captures object properties information for the object and compares it to a recorded baseline. <code>parameters\$ VP</code> is required; <code>ExpectedResult</code> and <code>Wait</code> are optional. |
| <code>recMethod\$</code> | Valid values: <ul style="list-style-type: none">▶ <code>Index=%</code>. The number of the parent or child object among all objects identified with the same base recognition method. Typically, <code>Index</code> is used after another recognition method qualifier — for example, <code>Name=\$; Index=%</code>.▶ <code>JavaText=\$</code>. A label that identifies the child object in the user interface.▶ <code>Name=\$</code>. A name that a developer assigns to a parent or child object to identify the object.▶ <code>Type=\$</code>. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The <code>Type</code> qualifier uses the following form: <code>Type=\$; recMethod=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>;\;</code>). <p>See <i>Recognition Methods in Java Commands</i> in Chapter 4 for other recognition methods that specify the parent object.</p> |





| Syntax Element | Description |
|---------------------------|---|
| <code>parameters\$</code> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <code>ExpectedResult=%</code>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> - <code>PASS</code>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail. - <code>FAIL</code>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail. ▶ <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points. ▶ <code>Wait=%,%</code>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10,40</code> (retry the test every 10 seconds, but time out the test after 40 seconds). |

Comments This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

If the parent object is not specified in the `recMethod` argument of this command, it must be specified in a preceding `Browser` command. For more information about specifying parent and child Java objects, see *Recognition Methods in Java Commands* in Chapter 4.

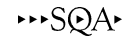
Example This example captures the properties of the Java panel named EmployeeList. The panel is located within the Java applet named Main. `JavaPanelVP` compares the properties to the recorded baseline in verification point `JPANELVP1`. At playback, the comparison is retried every 2 seconds and times out after 30 seconds.

```
Window SetContext, "Caption=Java demo", ""
Browser SetApplet, "Name=Main", ""
Result = JavaPanelVP (CompareProperties,
    "Type=JavaPanel;Name=EmployeeList", "VP=JPANELVP1;Wait=2,30")
```

See Also `JavaPanel`

JavaPopupMenu

User Action Command



Description Performs an action on a Java popup menu.

Syntax **JavaPopupMenu** *action%*, *recMethod\$*, *parameters\$*

| Syntax Element | Description |
|---------------------|--|
| <i>action%</i> | <p>The following action:</p> <ul style="list-style-type: none"> ▶ <i>MakeSelection</i>. Selects the specified item from a Java menu. ▶ <i>MouseClicked</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> must contain <i>Coords=x,y</i>. |
| <i>recMethod\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> ▶ <i>Index=%</i>. The number of the parent or child object among all objects identified with the same base recognition method. Typically, <i>Index</i> is used after another recognition method qualifier — for example, <i>Name=\$;Index=%</i>. ▶ <i>JavaText=\$</i>. A label that identifies the child object in the user interface. ▶ <i>Name=\$</i>. A name that a developer assigns to a parent or child object to identify the object. ▶ <i>Path=\$</i>. If <i>action%</i> is <i>MakeSelection</i>, the name of the popup menu and menu item. Sub-menus are separated by a pointer (->). ▶ <i>Type=\$</i>. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The <i>Type</i> qualifier uses the following form: <i>Type=\$;recMethod=\$</i>. Parent/child values are separated by a backslash and semicolons (; \ ;). <p>See <i>Recognition Methods in Java Commands</i> in Chapter 4 for other recognition methods that specify the parent object.</p> |
| <i>parameters\$</i> | <p>Valid value:</p> <ul style="list-style-type: none"> ▶ <i>Coords=x,y</i>. If <i>action%</i> is a mouse click, specifies the coordinates of the click, relative to the top left of the object. |

- Comments** Robot can recognize menus and sub-menus up to five levels deep.
- If the parent object is not specified in the *recMethod* argument of this command, it must be specified in a preceding *Browser* command. For more information about specifying parent and child Java objects, see *Recognition Methods in Java Commands* in Chapter 4.
- Example** This example opens the Java popup menu with a *Name* attribute of *PopupMenu1* and selects the *Open* option. The popup menu is located within the Java applet named *Main*.
- ```
Window SetContext, "Caption=Java demo", ""
Browser SetApplet, "Name=Main", ""
JavaPopupMenu MakeSelection, "Type=JavaPopupMenu; Index=1;
Path=PopupMenu1->Open", ""
```
- See Also**        JavaPopupMenuVP

## JavaPopupMenuVP

Verification Point Command



**Description**    Establishes a verification point for a Java popup menu.

**Syntax**         *Result* = **JavaPopupMenuVP**(*action%*, *recMethod\$*, *parameters\$*)

| Syntax Element     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i>     | The type of verification to perform. Valid value: <ul style="list-style-type: none"> <li>► <i>CompareProperties</i>. Captures object properties information for the object and compares it to a recorded baseline. <i>parameters\$ VP</i> is required; <i>ExpectedResult</i> and <i>Wait</i> are optional.</li> </ul>                                                                                                                                                                                                           |
| <i>recMethod\$</i> | Valid values: <ul style="list-style-type: none"> <li>► <i>Index=%</i>. The number of the parent or child object among all objects identified with the same base recognition method. Typically, <i>Index</i> is used after another recognition method qualifier — for example, <i>Name=\$; Index=%</i>.</li> <li>► <i>JavaText=\$</i>. A label that identifies the child object in the user interface.</li> <li>► <i>Name=\$</i>. A name that a developer assigns to a parent or child object to identify the object.</li> </ul> |

► ► ►



| Syntax Element            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|---------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                           | <ul style="list-style-type: none"> <li>▶ <code>Path=\$</code>. The name of the popup menu and menu item. Sub-menus are separated by a pointer ( <code>-&gt;</code> ).</li> <li>▶ <code>Type=\$</code>. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The <code>Type</code> qualifier uses the following form: <code>Type=\$;recMethod=\$</code>. Parent/child values are separated by a backslash and semicolons ( <code>;\;</code> ).</li> </ul> <p>See <i>Recognition Methods in Java Commands</i> in Chapter 4 for other recognition methods that specify the parent object.</p>                                                                                                                                                                                                                                                                                                                                                                                                    |
| <code>parameters\$</code> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>ExpectedResult=%</code>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> <li>- <code>PASS</code>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail.</li> <li>- <code>FAIL</code>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail.</li> </ul> </li> <li>▶ <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points.</li> <li>▶ <code>Wait=%,%</code>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10,40</code> (retry the test every 10 seconds, but time out the test after 40 seconds).</li> </ul> |

Comments This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

Robot can recognize menus and sub-menus up to five levels deep.

If the parent object is not specified in the `recMethod` argument of this command, it must be specified in a preceding `Browser` command. For more information about specifying parent and child Java objects, see *Recognition Methods in Java Commands* in Chapter 4.

**Example** This example captures the properties of the first Java popup menu in the applet (Index=1). The menu bar is located within the Java applet named Main. JavaPopupMenuVP compares the properties to the recorded baseline in verification point POPMENUVP1. At playback, the comparison is retried every 2 seconds and times out after 30 seconds.

```
Window SetContext, "Caption=Java demo", ""
Browser SetApplet, "Name=Main", ""
Result = JavaPopupMenuVP (CompareProperties,
 "Type=JavaPopupMenu;Index=1", "VP=POPMENUVP1;Wait=2,30")
```

**See Also** JavaPopupMenu

## JavaSplitPane

User Action Command



**Description** Performs an action on a Java split pane.

**Syntax** **JavaSplitPane** *action%*, *recMethod\$*, *parameters\$*

| Syntax Element     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i>     | <p>One of these mouse actions:</p> <ul style="list-style-type: none"> <li>▶ <i>MouseClicked</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> must contain <i>Coords=x,y</i>. See Appendix E for a list of mouse click values.</li> <li>▶ <i>ScrollAction</i>. One of these scroll actions:<br/> <i>HScrollTo</i>                      <i>VScrollTo</i><br/> <i>HScrollTo</i> and <i>VScrollTo</i> take the required parameter <i>Position=%</i>.</li> </ul> |
| <i>recMethod\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <i>Index=%</i>. The number of the parent or child object among all objects identified with the same base recognition method. Typically, <i>Index</i> is used after another recognition method qualifier — for example, <i>Name=\$;Index=%</i>.</li> <li>▶ <i>JavaText=\$</i>. A label that identifies the child object in the user interface.</li> </ul>                                                                                                                                                                                                           |

▶ ▶ ▶

## JavaSplitPaneVP

▶ ▶ ▶

| Syntax Element            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|---------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                           | <ul style="list-style-type: none"> <li>▶ <code>Name=\$</code>. A name that a developer assigns to a parent or child object to identify the object.</li> <li>▶ <code>Type=\$</code>. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The <code>Type</code> qualifier uses the following form: <code>Type=\$;recMethod=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>;\;</code>).</li> </ul> <p>See <i>Recognition Methods in Java Commands</i> in Chapter 4 for other recognition methods that specify the parent object.</p> |
| <code>parameters\$</code> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>Coords=x,y</code>. If <code>action%</code> is a mouse click, specifies the coordinates of the click, relative to the top left of the object.</li> <li>▶ <code>Position=%</code>. If <code>action%</code> is <code>VScrollTo</code> or <code>HScrollTo</code>, specifies the scroll bar value of the new scrolled-to position. Every scroll bar has an internal range and this parameter value is specific to that range.</li> </ul>                                                                                                                               |

**Comments** If the parent object is not specified in the `recMethod` argument of this command, it must be specified in a preceding `Browser` command. For more information about specifying parent and child Java objects, see *Recognition Methods in Java Commands* in Chapter 4.

**Example** This example clicks the Java split pane at coordinates 36, 25. The popup menu is located within the Java applet named Main.

```
Window SetContext, "Caption=Java demo", ""
Browser SetApplet, "Name=Main", ""
JavaSplitPane Click, "Type=JavaSplitPane;Name=SplitPane example",
 "Coords=36,25"
```

**See Also** `JavaSplitPaneVP`      `JavaSplitter`

## JavaSplitPaneVP

Verification Point Command

▶▶▶SQA▶

**Description** Establishes a verification point for a Java split pane.

**Syntax** `Result = JavaSplitPaneVP(action%,recMethod$,parameters$)`

| Syntax Element      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i>      | <p>The type of verification to perform. Valid value:</p> <ul style="list-style-type: none"> <li>▶ <code>CompareProperties</code>. Captures object properties information for the object and compares it to a recorded baseline. <i>parameters\$ VP</i> is required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <i>recMethod\$</i>  | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>Index=%</code>. The number of the parent or child object among all objects identified with the same base recognition method. Typically, <code>Index</code> is used after another recognition method qualifier — for example, <code>Name=\$; Index=%</code>.</li> <li>▶ <code>JavaText=\$</code>. A label that identifies the child object in the user interface.</li> <li>▶ <code>Name=\$</code>. A name that a developer assigns to a parent or child object to identify the object.</li> <li>▶ <code>Type=\$</code>. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The <code>Type</code> qualifier uses the following form: <code>Type=\$; recMethod=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>;\</code>).</li> </ul> <p>See <i>Recognition Methods in Java Commands</i> in Chapter 4 for other recognition methods that specify the parent object.</p>                    |
| <i>parameters\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>ExpectedResult=%</code>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> <li>– <code>PASS</code>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail.</li> <li>– <code>FAIL</code>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail.</li> </ul> </li> <li>▶ <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points.</li> <li>▶ <code>Wait=%, %</code>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10, 40</code> (retry the test every 10 seconds, but time out the test after 40 seconds).</li> </ul> |

## JavaSplitter

**Comments** This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

If the parent object is not specified in the *recMethod* argument of this command, it must be specified in a preceding Browser command. For more information about specifying parent and child Java objects, see *Recognition Methods in Java Commands* in Chapter 4.

**Example** This example captures the properties of the first Java split pane named SplitPane example. The split pane is located within the Java applet named Main. JavaSplitPaneVP compares the properties to the recorded baseline in verification point SPLITVP1. At playback, the comparison is retried every 2 seconds and times out after 30 seconds.

```
Window SetContext, "Caption=Java demo", ""
Browser SetApplet, "Name=Main", ""
Result = JavaSplitPaneVP (CompareProperties,
 "Type=JavaSplitPane;Name=SplitPane example",
 "VP=SPLITVP1;Wait=2,30")
```

**See Also** JavaSplitPane

## JavaSplitter

User Action Command

»»SQA»

**Description** Performs an action on a Java splitter.

**Syntax** **JavaSplitter** *action%*, *recMethod\$*, *parameters\$*

| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i> | <p>Valid values:</p> <ul style="list-style-type: none"><li>► <i>MouseClicked</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> must contain <i>Coords=x,y</i>. See Appendix E for a list of mouse click values.</li><li>► <i>ScrollAction</i>. One of these scroll actions:<br/>HScrollTo                      VScrollTo<br/>HScrollTo and VScrollTo take the required parameter <i>Position=%</i>.</li></ul> |

► ► ►

▶ ▶ ▶

| Syntax Element       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>recMethod</i> \$  | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <i>Index</i>=%. The number of the parent or child object among all objects identified with the same base recognition method. Typically, <i>Index</i> is used after another recognition method qualifier — for example, <i>Name</i>=<i>\$</i> ; <i>Index</i>=%.</li> <li>▶ <i>Name</i>=<i>\$</i>. A name that a developer assigns to a parent or child object to identify the object.</li> <li>▶ <i>Type</i>=<i>\$</i>. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The <i>Type</i> qualifier uses the following form: <i>Type</i>=<i>\$</i> ; <i>recMethod</i>=<i>\$</i>. Parent/child values are separated by a backslash and semicolons (<i>;</i> \ <i>;</i>).</li> </ul> <p>See <i>Recognition Methods in Java Commands</i> in Chapter 4 for other recognition methods that specify the parent object.</p> |
| <i>parameters</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <i>Coords</i>=<i>x,y</i>. If <i>action</i>% is a mouse click, specifies the coordinates of the click, relative to the top left of the object.</li> <li>▶ <i>Position</i>=%. If <i>action</i>% is <i>VScrollTo</i> or <i>HScrollTo</i>, specifies the scroll bar value of the new scrolled-to position. Every scroll bar has an internal range and this parameter value is specific to that range.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

Comments      *JavaSplitter* acts on the splitter object itself. *JavaSplitPane* relies on the split pane to perform the splitter action.

If the parent object is not specified in the *recMethod* argument of this command, it must be specified in a preceding *Browser* command. For more information about specifying parent and child Java objects, see *Recognition Methods in Java Commands* in Chapter 4.

Example        This example sets the scroll-to position of a Java splitter component as 234.

```

Sub Main
 Dim Result As Integer
 'Initially Recorded: 06/09/99 14:15:21
 'Script Name: JavaSplitter
 Window SetContext, "Caption=Project1", ""
 JavaSplitter VScrollTo,
 "JavaCaption=Project1;\;Type=JavaSplitter;Index=1",
 "Position=234"
 Window SetTestContext, "Caption=Project1", ""
 Result = JavaSplitterVP (CompareProperties,
 "JavaCaption=Project1;\;Type=JavaSplitter;Index=1",
 "VP=Object Properties")
 Window ResetTestContext, "", ""
End Sub

```

## JavaSplitterVP

### Verification Point Command



**Description**    Establishes a verification point for a Java splitter.

**Syntax**    `Result = JavaSplitterVP(action%, recMethod$, parameters$)`

| Syntax Element      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i>      | <p>The type of verification to perform. Valid value:</p> <ul style="list-style-type: none"> <li>▶ <code>CompareProperties</code>. Captures object properties information for the object and compares it to a recorded baseline. <i>parameters\$</i> VP is required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <i>recMethod\$</i>  | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>Index=%</code>. The number of the parent or child object among all objects identified with the same base recognition method. Typically, <code>Index</code> is used after another recognition method qualifier — for example, <code>Name=\$; Index=%</code>.</li> <li>▶ <code>Name=\$</code>. A name that a developer assigns to a parent or child object to identify the object.</li> <li>▶ <code>Type=\$</code>. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The <code>Type</code> qualifier uses the following form: <code>Type=\$; recMethod=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>;\;</code>).</li> </ul> <p>See <i>Recognition Methods in Java Commands</i> in Chapter 4 for other recognition methods that specify the parent object.</p> |
| <i>parameters\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>ExpectedResult=%</code>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> <li>– <code>PASS</code>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail.</li> <li>– <code>FAIL</code>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail.</li> </ul> </li> </ul>                                                                                                                                                                                                                                                 |







| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                            |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                | <ul style="list-style-type: none"> <li>▶ VP=\$. The verification point ID. IDs must be unique within a script. Required for all verification points.</li> <li>▶ Wait=%,%. A Wait State that specifies the verification point's Retry value and a Timeout value, as in Wait=10,40 (retry the test every 10 seconds, but time out the test after 40 seconds).</li> </ul> |

**Comments** This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

If the parent object is not specified in the *recMethod* argument of this command, it must be specified in a preceding **Browser** command. For more information about specifying parent and child Java objects, see *Recognition Methods in Java Commands* in Chapter 4.

**Example** The following example establishes an object properties verification point for a Java splitter component.

```

Sub Main
 Dim Result As Integer

 'Initially Recorded: 06/09/99 14:15:21
 'Script Name: JavaSplitter

 Window SetContext, "Caption=Project1", ""
 JavaSplitter VScrollTo,
 "JavaCaption=Project1;\;Type=JavaSplitter;Index=1",
 "Position=234"
 Window SetTestContext, "Caption=Project1", ""
 Result = JavaSplitterVP (CompareProperties,
 "JavaCaption=Project1;\;Type=JavaSplitter;Index=1",
 "VP=Object Properties")
 Window ResetTestContext, "", ""

End Sub

```

**See Also** JavaSplitter

## JavaTable

User Action Command

»»SQAS»

Description Performs an action on a Java table.

Syntax **JavaTable** *action%*, *recMethod\$*, *parameters\$*

| Syntax Element     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i>     | <p>One of these mouse actions:</p> <ul style="list-style-type: none"> <li>▶ <i>MouseClicked</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> must contain <i>Coords=x,y</i>.</li> <li>▶ <i>MouseDown</i>. The dragging of the mouse while mouse buttons and/or shifting keys (Ctrl, Alt, Shift) are pressed. When <i>action%</i> contains a mouse-drag value, <i>parameters\$</i> must contain <i>Coords=x1,y1,x2,y2</i>.</li> </ul> <p>See Appendix E for a list of mouse click and drag values.</p>                                                                                                                                                                                                                                                                                                           |
| <i>recMethod\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <i>Index=%</i>. The number of the parent or child object among all objects identified with the same base recognition method. Typically, <i>Index</i> is used after another recognition method qualifier — for example, <i>Name=\$;Index=%</i>.</li> <li>▶ <i>JavaText=\$</i>. A label that identifies the child object in the user interface.</li> <li>▶ <i>Name=\$</i>. A name that a developer assigns to a parent or child object to identify the object.</li> <li>▶ <i>Type=\$</i>. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The <i>Type</i> qualifier uses the following form: <i>Type=\$;recMethod=\$</i>. Parent/child values are separated by a backslash and semicolons (<i>;\</i>).</li> </ul> <p>See <i>Recognition Methods in Java Commands</i> in Chapter 4 for other recognition methods that specify the parent object.</p> |

▶ ▶ ▶



| Syntax Element       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>parameters</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <i>Coords=x,y</i>. If <i>action%</i> is a mouse click in a writeable cell, specifies the coordinates of the click, relative to the top left of the cell being acted upon.</li> <li>▶ <i>Coords=x1,y1,x2,y2</i>. If <i>action%</i> is a mouse drag in a writeable cell, specifies the coordinates, where <i>x1,y1</i> are the starting coordinates of the drag, and <i>x2,y2</i> are the ending coordinates. The coordinates are relative to the top left of the cell being acted upon.</li> <li>▶ <i>Col=%</i>. Identifies the index of a column in the table.</li> <li>▶ <i>ColTitle=\$</i>. Identifies the title of the table column.</li> <li>▶ <i>EndCol= %</i>. Identifies the index of the ending column in the table.</li> <li>▶ <i>EndColTitle=\$</i>. Identifies the title of the ending column for a <i>MouseDown</i> action.</li> <li>▶ <i>Row=%</i>. Identifies the index of a row in the table.</li> <li>▶ <i>StartCol= %</i>. Identifies the index of the starting column.</li> <li>▶ <i>StartColTitle=\$</i>. Identifies the title of the starting column.</li> <li>▶ <i>Text=\$</i>. Identifies the text of an item in the table.</li> <li>▶ <i>Value=%</i>. The current value of the table item.</li> </ul> |

**Comments** If the parent object is not specified in the *recMethod* argument of this command, it must be specified in a preceding *Browser* command. For more information about specifying parent and child Java objects, see *Recognition Methods in Java Commands* in Chapter 4.

**Example** This example clicks the first table in the Java applet named Main. The click occurs in the column titled Favorite Number at coordinates 36, 10. The value is 2.

```
Window SetContext, "Caption=Java demo", ""
Browser SetApplet, "Name=Main", ""
JavaTable Click, "Type=JavaTable;Index=1",
 "StartColTitle=LastName;ColTitle=FavoriteNumber;Value=2;
 Coords=36,10"
```

**See Also** JavaTableVP

## JavaTableVP

Verification Point Command

**Description** Establishes a verification point for a Java table.

**Syntax** `Result = JavaTableVP (action%, recMethod$, parameters$)`

| Syntax Element      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|---------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i>      | <p>The type of verification to perform. Valid value:</p> <ul style="list-style-type: none"> <li>▶ <code>CompareProperties</code>. Captures object properties information for the object and compares it to a recorded baseline. <i>parameters\$</i> VP is required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <i>recMethod\$</i>  | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>Index=%</code>. The number of the parent or child object among all objects identified with the same base recognition method. Typically, <code>Index</code> is used after another recognition method qualifier — for example, <code>Name=\$; Index=%</code>.</li> <li>▶ <code>JavaText=\$</code>. A label that identifies the child object in the user interface.</li> <li>▶ <code>Name=\$</code>. A name that a developer assigns to a parent or child object to identify the object.</li> <li>▶ <code>Type=\$</code>. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The <code>Type</code> qualifier uses the following form: <code>Type=\$; recMethod=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>;\</code>).</li> </ul> <p>See <i>Recognition Methods in Java Commands</i> in Chapter 4 for other recognition methods that specify the parent object.</p> |
| <i>parameters\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>ExpectedResult=%</code>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> <li>– <code>PASS</code>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail.</li> <li>– <code>FAIL</code>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail.</li> </ul> </li> </ul>                                                                                                                                                                                                                                                                                                                                                     |

▶ ▶ ▶

▶ ▶ ▶

| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                                                                   |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                | <ul style="list-style-type: none"> <li>▶ <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points.</li> <li>▶ <code>Wait=%,%</code>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10,40</code> (retry the test every 10 seconds, but time out the test after 40 seconds).</li> </ul> |

**Comments** This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

If the parent object is not specified in the *recMethod* argument of this command, it must be specified in a preceding *Browser* command. For more information about specifying parent and child Java objects, see *Recognition Methods in Java Commands* in Chapter 4.

**Example** This example captures the properties of the Java table named EmployeeList. The table is located within the Java applet named Main. JavaTableVP compares the properties to the recorded baseline in verification point TABLEVP1. At playback, the comparison is retried every 2 seconds and times out after 30 seconds.

```
Window SetContext, "Caption=Java demo", ""
Browser SetApplet, "Name=Main", ""
Result = JavaTableVP (CompareProperties,
 "Type=JavaTable;Name=EmployeeList", "VP=TABLEVP1;Wait=2,30")
```

**See Also** JavaTable

## JavaTableHeader

User Action Command



**Description** Performs an action on a Java table header.

**Syntax** `JavaTableHeader action%, recMethod$, parameters$`

| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                        |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i> | <p>One of these mouse actions:</p> <ul style="list-style-type: none"> <li>▶ <i>MouseClicked</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> must contain <code>Coords=x,y</code>.</li> </ul> |

▶ ▶ ▶

## JavaTableHeader

▶ ▶ ▶

| Syntax Element      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|---------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                     | <ul style="list-style-type: none"> <li>▶ <i>MouseDown</i>. The dragging of the mouse while mouse buttons and/or shifting keys (Ctrl, Alt, Shift) are pressed. When <i>action%</i> contains a mouse-drag value, <i>parameters\$</i> must contain <i>Coords=x1,y1,x2,y2</i>.<br/>See Appendix E for a list of mouse click and drag values.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <i>recMethod\$</i>  | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <i>Index=%</i>. The number of the parent or child object among all objects identified with the same base recognition method. Typically, <i>Index</i> is used after another recognition method qualifier — for example, <i>Name=\$; Index=%</i>.</li> <li>▶ <i>JavaText=\$</i>. A label that identifies the child object in the user interface.</li> <li>▶ <i>Name=\$</i>. A name that a developer assigns to a parent or child object to identify the object.</li> <li>▶ <i>Type=\$</i>. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The <i>Type</i> qualifier uses the following form: <i>Type=\$; recMethod=\$</i>. Parent/child values are separated by a backslash and semicolons (<i>;\</i>).</li> </ul> <p>See <i>Recognition Methods in Java Commands</i> in Chapter 4 for other recognition methods that specify the parent object.</p> |
| <i>parameters\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <i>Coords=x1,y1,x2,y2</i>. If <i>action%</i> is a mouse drag, specifies the coordinates, where <i>x1,y1</i> are the starting coordinates of the drag, and <i>x2,y2</i> are the ending coordinates. The coordinates are relative to the top left of the specified start column header cell.</li> <li>▶ <i>Col=%</i>. Identifies the index of a column in the table.</li> <li>▶ <i>ColTitle=\$</i>. Identifies the title of the table column.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |

**Comments** If the parent object is not specified in the *recMethod* argument of this command, it must be specified in a preceding *Browser* command. For more information about specifying parent and child Java objects, see *Recognition Methods in Java Commands* in Chapter 4.

**Example** This example clicks the table column header title Employee Number in a table named EmployeeList. The table is located within the Java applet named Main.

```
Window SetContext, "Caption=Java demo", ""
Browser SetApplet, "Name=Main", ""
JavaTableHeader Click, "Type=JavaTable;Name=EmployeeList",
"ColTitle=Employee Number"
```

See Also     JavaTable  
               JavaTableVP  
               JavaTableHeaderVP

## JavaTableHeaderVP

Verification Point Command

▶▶SQA▶

Description   Establishes a verification point for a Java table header.

Syntax        *Result* = **JavaTableHeaderVP** (*action%*, *recMethod\$*, *parameters\$*)

| Syntax Element     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i>     | <p>The type of verification to perform. Valid value:</p> <ul style="list-style-type: none"> <li>▶ CompareProperties. Captures object properties information for the object and compares it to a recorded baseline. <i>parameters\$</i> VP is required; ExpectedResult and Wait are optional.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <i>recMethod\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ Index=%. The number of the parent or child object among all objects identified with the same base recognition method. Typically, Index is used after another recognition method qualifier — for example, Name=\$; Index=%.</li> <li>▶ JavaText=\$. A label that identifies the child object in the user interface.</li> <li>▶ Name=\$. A name that a developer assigns to a parent or child object to identify the object.</li> <li>▶ Type=\$. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The Type qualifier uses the following form: Type=\$; recMethod=\$. Parent/child values are separated by a backslash and semicolons (; \ ;).</li> </ul> <p>See <i>Recognition Methods in Java Commands</i> in Chapter 4 for other recognition methods that specify the parent object.</p> |

▶ ▶ ▶

## JavaTableHeaderVP



| Syntax Element            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|---------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>parameters\$</code> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>ExpectedResult=%</code>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> <li>– <code>PASS</code>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail.</li> <li>– <code>FAIL</code>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail.</li> </ul> </li> <li>▶ <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points.</li> <li>▶ <code>Wait=%,%</code>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10,40</code> (retry the test every 10 seconds, but time out the test after 40 seconds).</li> </ul> |

**Comments** This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

If the parent object is not specified in the `recMethod` argument of this command, it must be specified in a preceding `Browser` command. For more information about specifying parent and child Java objects, see *Recognition Methods in Java Commands* in Chapter 4.

**Example** This example captures the properties of the Java table header named `EmployeeList`. The table is located within the Java applet named `Main`. `JavaTableHeaderVP` compares the properties to the recorded baseline in verification point `TABLEHEADERVP1`. At playback, the comparison is retried every 2 seconds and times out after 30 seconds.

```
Window SetContext, "Caption=Java demo", ""
Browser SetApplet, "Name=Main", ""
Result=JavaTableHeaderVP (CompareProperties,
 "Type=JavaTableHeader; Name=EmployeeList",
 "VP=TABLEHEADERVP1;Wait=2,30")
```

**See Also** `JavaTable`  
`JavaTableHeader`



# JavaTree

User Action Command

▶▶SQA▶

**Description** Performs an action on a Java tree component.

**Syntax** **JavaTree** *action%*, *recMethod\$*, *parameters\$*

| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i> | <p>One of these mouse actions:</p> <ul style="list-style-type: none"> <li>▶ <b>Collapse.</b> Collapses the tree. <i>recMethod\$</i> must contain one of the Java recognition methods, and <i>parameters\$</i> must contain either <code>Text</code> or <code>JavaRow</code>.</li> <li>▶ <b>Deselect.</b> Deselects the specified item from an extended Java tree component in <code>multipleMode</code>. <i>recMethod\$</i> must contain one of the Java recognition methods, and <i>parameters\$</i> must contain either <code>Text</code> or <code>JavaRow</code>.</li> <li>▶ <b>Expand.</b> Expands the tree. <i>recMethod\$</i> must contain one of the Java recognition methods, and <i>parameters\$</i> must contain either <code>Text</code> or <code>JavaRow</code>.</li> <li>▶ <b>ExtendSelection.</b> Selects the specified item from an extended Java tree component in <code>multipleMode</code>. <i>recMethod\$</i> must contain one of the Java recognition methods, and <i>parameters\$</i> must contain either <code>Text</code> or <code>JavaRow</code>.</li> <li>▶ <b>MakeSelection.</b> Selects the specified item in a Java tree. <i>recMethod\$</i> must contain one of the Java recognition methods, and <i>parameters\$</i> must contain either <code>Text</code> or <code>JavaRow</code>.</li> <li>▶ <b>MouseClicked.</b> The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> must contain <code>Coords=x,y</code>.</li> </ul> <p>See Appendix E for a list of mouse click values.</p> |

▶ ▶ ▶



| Syntax Element            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|---------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>recMethod\$</code>  | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>Index=%</code>. The number of the parent or child object among all objects identified with the same base recognition method. Typically, <code>Index</code> is used after another recognition method qualifier — for example, <code>Name=\$ ; Index=%</code>.</li> <li>▶ <code>JavaText=\$</code>. A label that identifies the child object in the user interface.</li> <li>▶ <code>Name=\$</code>. A name that a developer assigns to a parent or child object to identify the object.</li> <li>▶ <code>Type=\$</code>. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The <code>Type</code> qualifier uses the following form: <code>Type=\$ ; recMethod=\$</code>. Parent/child values are separated by a backslash and semicolons (<code> ; \ ;</code>).</li> </ul> <p>See <i>Recognition Methods in Java Commands</i> in Chapter 4 for other recognition methods that specify the parent object.</p> |
| <code>parameters\$</code> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>Coords=x,y</code>. If <code>action%</code> is a mouse click, specifies the coordinates of the click, relative to the top left of the object.</li> <li>▶ <code>JavaRow=%</code>. Identifies the row number of an item in the list.</li> <li>▶ <code>Text=\$</code>. Identifies the text of an item in the list. The tree items are separated by a pointer (<code> -&gt; </code>).</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

**Comments** If the parent object is not specified in the `recMethod` argument of this command, it must be specified in a preceding `Browser` command. For more information about specifying parent and child Java objects, see *Recognition Methods in Java Commands* in Chapter 4.

**Example** This example expands the Jazz node of the Java tree with a Name attribute of Music. The menu bar is located within the Java applet named Main.

```
Window SetContext, "Caption=Java demo", ""
Browser SetApplet, "Name=Main", ""
JavaTree Expand, "Type=JavaTree;Name=Music", "Text=Music->Jazz"
```

**See Also** JavaTreeVP

# JavaTreeVP

Verification Point Command

▶▶SQA▶

**Description** Establishes a verification point for a Java tree component.

**Syntax** `Result = JavaTreeVP (action%, recMethod$, parameters$)`

| Syntax Element      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|---------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i>      | <p>The type of verification to perform. Valid value:</p> <ul style="list-style-type: none"> <li>▶ <code>CompareProperties</code>. Captures object properties information for the object and compares it to a recorded baseline. <i>parameters\$</i> VP is required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <i>recMethod\$</i>  | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>Index=%</code>. The number of the parent or child object among all objects identified with the same base recognition method. Typically, <code>Index</code> is used after another recognition method qualifier — for example, <code>Name=\$; Index=%</code>.</li> <li>▶ <code>JavaText=\$</code>. A label that identifies the child object in the user interface.</li> <li>▶ <code>Name=\$</code>. A name that a developer assigns to a parent or child object to identify the object.</li> <li>▶ <code>Type=\$</code>. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The <code>Type</code> qualifier uses the following form: <code>Type=\$; recMethod=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>;\</code>).</li> </ul> <p>See <i>Recognition Methods in Java Commands</i> in Chapter 4 for other recognition methods that specify the parent object.</p> |
| <i>parameters\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>ExpectedResult=%</code>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> <li>– <code>PASS</code>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail.</li> <li>– <code>FAIL</code>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail.</li> </ul> </li> </ul>                                                                                                                                                                                                                                                                                                                                                     |

▶ ▶ ▶

## JavaWindow

▶ ▶ ▶

| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                                                                |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                | <ul style="list-style-type: none"><li>▶ <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points.</li><li>▶ <code>Wait=%,%</code>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10,40</code> (retry the test every 10 seconds, but time out the test after 40 seconds).</li></ul> |

**Comments** This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

If the parent object is not specified in the *recMethod* argument of this command, it must be specified in a preceding `Browser` command. For more information about specifying parent and child Java objects, see *Recognition Methods in Java Commands* in Chapter 4.

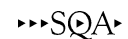
**Example** This example captures the properties of the Java tree with a Name attribute of `JavaTree1`. The tree is located within the Java applet named `Main`. `JavaTreeVP` compares the properties to the recorded baseline in verification point `TREEVP1`. At playback, the comparison is retried every 2 seconds and times out after 30 seconds.

```
Window SetContext, "Caption=Java demo", ""
Browser SetApplet, "Name=Main", ""
Result = JavaTreeVP (CompareProperties,
 "Type=JavaTree;Name=JavaTree1", "VP=TREEVP1;Wait=2,30")
```

**See Also** `JavaTree`

## JavaWindow

User Action Command



**Description** Performs an action on a Java window.

**Syntax** `JavaWindow action%, recMethod$, parameters$`

| Syntax Element      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i>      | <p>One of these mouse actions:</p> <ul style="list-style-type: none"> <li>▶ <i>MouseClicked</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> must contain <code>Coords=x,y</code>.</li> <li>▶ <i>MouseDown</i>. The dragging of the mouse while mouse buttons and/or shifting keys (Ctrl, Alt, Shift) are pressed. When <i>action%</i> contains a mouse-drag value, <i>parameters\$</i> must contain <code>Coords=x1,y1,x2,y2</code>.</li> </ul> <p>See Appendix E for a list of mouse click and drag values.</p>                                                                                                                                                                                                                                                                                                                                                      |
| <i>recMethod\$</i>  | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>Index=%</code>. The number of the parent or child object among all objects identified with the same base recognition method. Typically, <code>Index</code> is used after another recognition method qualifier — for example, <code>Name=\$;Index=%</code>.</li> <li>▶ <code>JavaText=\$</code>. A label that identifies the child object in the user interface.</li> <li>▶ <code>Name=\$</code>. A name that a developer assigns to a parent or child object to identify the object.</li> <li>▶ <code>Type=\$</code>. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The <code>Type</code> qualifier uses the following form: <code>Type=\$;recMethod=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>;\;</code>).</li> </ul> <p>See <i>Recognition Methods in Java Commands</i> in Chapter 4 for other recognition methods that specify the parent object.</p> |
| <i>parameters\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>Coords=x,y</code>. If <i>action%</i> is a mouse click, specifies the coordinates of the click, relative to the top left of the object.</li> <li>▶ <code>Coords=x1,y1,x2,y2</code>. If <i>action%</i> is a mouse drag, specifies the coordinates, where <code>x1,y1</code> are the starting coordinates of the drag, and <code>x2,y2</code> are the ending coordinates. The coordinates are relative to the top left of the object.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

Comments     If the parent object is not specified in the *recMethod* argument of this command, it must be specified in a preceding *Browser* command. For more

JavaWindowVP

information about specifying parent and child Java objects, see *Recognition Methods in Java Commands* in Chapter 4.

**Example** This example clicks the window titled EmployeeList. The window is located within the Java applet named Main.

```
Window SetContext, "Caption=Java demo", ""
Browser SetApplet, "Name=Main", ""
JavaWindow Click, "Type=JavaWindow;Name=EmployeeList", "Coords=25,50"
```

**See Also** JavaWindowVP

## JavaWindowVP

Verification Point Command



**Description** Establishes a verification point for a Java window.

**Syntax** `Result = JavaWindowVP (action%, recMethod$, parameters$)`

| Syntax Element           | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>action%</code>     | The type of verification to perform. Valid value: <ul style="list-style-type: none"><li>▶ <code>CompareProperties</code>. Captures object properties information for the object and compares it to a recorded baseline. <code>parameters\$ VP</code> is required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li></ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <code>recMethod\$</code> | Valid values: <ul style="list-style-type: none"><li>▶ <code>Index=%</code>. The number of the parent or child object among all objects identified with the same base recognition method. Typically, <code>Index</code> is used after another recognition method qualifier — for example, <code>Name=\$; Index=%</code>.</li><li>▶ <code>JavaText=\$</code>. A label that identifies the child object in the user interface.</li><li>▶ <code>Name=\$</code>. A name that a developer assigns to a parent or child object to identify the object.</li><li>▶ <code>Type=\$</code>. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The <code>Type</code> qualifier uses the following form: <code>Type=\$; recMethod=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>;\;</code>).</li></ul> <p>See <i>Recognition Methods in Java Commands</i> in Chapter 4 for other recognition methods that specify the parent object.</p> |





| Syntax Element            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|---------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>parameters\$</code> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>ExpectedResult=%</code>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> <li>– <code>PASS</code>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail.</li> <li>– <code>FAIL</code>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail.</li> </ul> </li> <li>▶ <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points.</li> <li>▶ <code>Wait=%,%</code>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10,40</code> (retry the test every 10 seconds, but time out the test after 40 seconds).</li> </ul> |

**Comments** This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

If the parent object is not specified in the `recMethod` argument of this command, it must be specified in a preceding `Browser` command. For more information about specifying parent and child Java objects, see *Recognition Methods in Java Commands* in Chapter 4.

**Example** This example captures the properties of the Java window named `EmployeeList`. The table is located within the Java applet named `Main`. `JavaWindowVP` compares the properties to the recorded baseline in verification point `JAWINDOWVP1`. At playback, the comparison is retried every 2 seconds and times out after 30 seconds.

```
Window SetContext, "Caption=Java demo", ""
Browser SetApplet, "Name=Main", ""
Result = JavaWindowVP (CompareProperties,
 "Type=JavaWindow;Name=EmployeeList", "VP=JAWINDOWVP1;Wait=2,30")
```

**See Also** `JavaWindow`

Kill

## Kill

Statement

---

Description    Deletes files from a hard disk or diskette.

Syntax        **kill** *pathname\$*

| Syntax Element    | Description                                                  |
|-------------------|--------------------------------------------------------------|
| <i>pathname\$</i> | An expression that specifies a valid DOS file specification. |

Comments      The *pathname\$* specification can contain paths and wildcards. **kill** deletes files only, not directories. Use the **Rmdir** function to delete directories.

Example        This example prompts a user for an account number, opens a file, searches for the account number and displays the matching letter for that number. The second sub procedure, **CREATEFILE**, creates the file **C:\TEMP001** used by the main sub procedure. After processing is complete, the first sub procedure uses **kill** to delete the file.

```
Declare Sub createfile()
Global x as Integer
Global y(100) as String
Sub main
 Dim acctno as Integer
 Dim msgtext
 Call createfile
 i: acctno=InputBox("Enter an account number from 1-10:")
 If acctno<1 Or acctno>10 then
 MsgBox "Invalid account number. Try again."
 Goto i:
 End if
 x=1
 Open "C:\TEMP001" for Input as #1
 Do Until x=acctno
 Input #1, x,y(x)
 Loop
 msgtext="The letter for account number " & x & " is: " & y(x)
 Close #1
 MsgBox msgtext
 kill "C:\TEMP001"
End Sub
Sub createfile()
' Put the numbers 1-10 and letters A-J into a file
 Dim startletter
 Open "C:\TEMP001" for Output as #1
 startletter=65
 For x=1 to 10
 y(x)=Chr(startletter)
 startletter=startletter+1
 Next x
 For x=1 to 10
 Write #1, x,y(x)
 Next x
 Close #1
End Sub
```



See Also      `FileAttr`            `GetAttr`  
                  `FileDateTime`      `Rmdir`

## Label

User Action Command

»»»SQA»

**Description**      Performs an action on a label control.

**Syntax**            **Label** *action%*, *recMethod\$*

| Syntax Element     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i>     | <p>The following mouse action:</p> <ul style="list-style-type: none"> <li>▶ <i>MouseClick</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). Does not require coordinate information.</li> </ul> <p>See Appendix E for a list of mouse click values.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <i>recMethod\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <i>ID=%</i>. The object's internal Windows ID.</li> <li>▶ <i>Index=%</i>. The number of the object among all objects identified with the same base recognition method. Typically, <i>Index</i> is used after another recognition method qualifier — for example, <i>Name=\$; Index=%</i>.</li> <li>▶ <i>JavaText=\$</i>. A label that identifies the object in the user interface.</li> <li>▶ <i>Name=\$</i>. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be <i>Command1</i>.</li> <li>▶ <i>ObjectIndex=%</i>. The number of the object among all objects of the same type in the same window.</li> <li>▶ <i>State=\$</i>. An optional qualifier for any other recognition method. There are two possible values for this setting: <i>Enabled</i> and <i>Disabled</i>. The default state is the state of the current context window (as set in the most recent <i>Window SetContext</i> command), or <i>Enabled</i> if the state has not been otherwise declared.</li> <li>▶ <i>Text=\$</i>. The text displayed on the object.</li> </ul> |

▶ ▶ ▶

## LabelVP

▶ ▶ ▶

| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                | <ul style="list-style-type: none"><li>▶ <code>Type=\$</code>. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The <code>Type</code> qualifier uses the following form: <code>Type=\$;recMethod=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>;\</code>).</li><li>▶ <code>VisualText=\$</code>. An optional setting used to identify an object by its visible text. It is for user clarification only and does not affect object recognition.</li></ul> |

Comments None.

Example This example clicks the label identified with the text Tuesday, March 12, 1999.

```
Label Click, "Text=Tuesday, March 12, 1999"
```

See Also  
CheckBox  
PushButton  
RadioButton

## LabelVP

Verification Point Command

▶▶▶SQA▶

Description Establishes a verification point for a label control.

Syntax `Result = LabelVP (action%, recMethod$, parameters$)`

| Syntax Element       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>action%</code> | <p>The type of verification to perform. Valid values:</p> <ul style="list-style-type: none"><li>▶ <code>CompareNumeric</code>. Captures the numeric value of the text of the object and compares it to the value of <code>parameters\$ Value or Range</code>. <code>parameters\$ VP</code> and either <code>Value or Range</code> are required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li><li>▶ <code>CompareProperties</code>. Captures object properties information for the object and compares it to a recorded baseline. <code>parameters\$ VP</code> is required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li></ul> |

▶ ▶ ▶

▶ ▶ ▶

| Syntax Element       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                      | <ul style="list-style-type: none"> <li>▶ <code>CompareText</code>. Captures the text of the object and compares it to a recorded baseline. <i>parameters</i>\$ VP and <code>Type</code> are required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li> <li>▶ <code>VerifyIsBlank</code>. Checks that the object has no text. <i>parameters</i>\$ VP is required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <i>recMethod</i> \$  | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>ID=%</code>. The object's internal Windows ID.</li> <li>▶ <code>Index=%</code>. The number of the object among all objects identified with the same base recognition method. Typically, <code>Index</code> is used after another recognition method qualifier — for example, <code>Name=\$; Index=%</code>.</li> <li>▶ <code>JavaText=\$</code>. A label that identifies the object in the user interface.</li> <li>▶ <code>Name=\$</code>. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a <code>command</code> button might be <code>Command1</code>.</li> <li>▶ <code>ObjectIndex=%</code>. The number of the object among all objects of the same type in the same window.</li> <li>▶ <code>Text=\$</code>. The text displayed on the object.</li> <li>▶ <code>Type=\$</code>. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The <code>Type</code> qualifier uses the following form: <code>Type=\$; recMethod=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>;\;</code>).</li> </ul> |
| <i>parameters</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>ExpectedResult=%</code>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> <li>– <code>PASS</code>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail.</li> <li>– <code>FAIL</code>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail.</li> </ul> </li> <li>▶ <code>Range=&amp;, &amp;</code>. Used with the action <code>CompareNumeric</code> when a numeric range comparison is being performed, as in <code>Range=2, 12</code> (test for numbers in this range). The values are inclusive.</li> </ul>                                                                                                                                                                                                                                                                                                                                    |

▶ ▶ ▶

## LabelVP

▶ ▶ ▶

| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                | <ul style="list-style-type: none"> <li>▶ <code>Type=\$</code>. Specifies the verification method to use for <code>CompareText</code> actions. The possible values are: <code>CaseSensitive</code>, <code>CaseInsensitive</code>, <code>FindSubStr</code>, <code>FindSubStrI</code> (case insensitive), and <code>UserDefined</code>. See <i>Comments</i> for more information. If <code>UserDefined</code> is specified, two additional parameters are required:               <ul style="list-style-type: none"> <li>- <code>DLL=\$</code>. The full path and file name of the library that contains the function.</li> <li>- <code>Function=\$</code>. The name of the custom function to use in comparing the text.</li> </ul> </li> <li>▶ <code>Value=&amp;</code>. Used with the action <code>CompareNumeric</code> when a numeric equivalence comparison is being performed, as in <code>Value=25</code> (test against the value 25).</li> <li>▶ <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points.</li> <li>▶ <code>Wait=% , %</code>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10 , 40</code> (retry the test every 10 seconds, but time out the test after 40 seconds).</li> </ul> |

**Comments** This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

With the `Type=$` parameter, `CaseSensitive` and `CaseInsensitive` require a full match between the current baseline text and the text captured during playback. With `FindSubStr` and `FindSubStrI`, the current baseline can be a substring of the text captured during playback. The substring can appear anywhere in the playback text. To modify the current baseline text, double-click the verification point name in the Robot Asset pane (to the left of the script).

**Example** This example captures the text of the second label object in the window `ObjectIndex=2` and performs a case-sensitive comparison with the recorded baseline in verification point `VPTRIAL`.

```
Result = LabelVP (CompareText, "ObjectIndex=2",
 "VP=VPTRIAL;Type=CaseSensitive")
```

**See Also**

|            |           |
|------------|-----------|
| ComboBoxVP | TextBoxVP |
| ComboBoxVP | ListboxVP |

## LBound

### Function

---

**Description** Returns the lower bound of the subscript range for the specified array.

**Syntax** **LBound**(*arrayname* [, *dimension* ])

| Syntax Element   | Description                   |
|------------------|-------------------------------|
| <i>arrayname</i> | The name of the array to use. |
| <i>dimension</i> | The dimension to use.         |

**Comments** The dimensions of an array are numbered starting with 1. If the *dimension* is not specified, 1 is used as a default.

LBound can be used with UBound to determine the length of an array.

**Example** This example resizes an array if the user enters more data than can fit in the array. It uses LBound and UBound to determine the existing size of the array and ReDim to resize it. Option Base sets the default lower bound of the array to 1.

```
Option Base 1
Sub main
 Dim arrayvar() as Integer
 Dim count as Integer
 Dim answer as String
 Dim x, y as Integer
 Dim total
 total=0
 x=1
 count=InputBox("How many test scores do you have?")
 ReDim arrayvar(count)
start:
 Do until x=count+1
 arrayvar(x)=InputBox("Enter test score #" &x & ":")
 x=x+1
 Loop
 answer=InputBox$("Do you have more scores? (Y/N)")
 If answer="Y" or answer="y" then
 count=InputBox("How many more do you have?")
 If count<>0 then
 count=count+(x-1)
 ReDim Preserve arrayvar(count)
 Goto start
 End If
 End If
 x=LBound(arrayvar,1)
 count=UBound(arrayvar,1)
 For y=x to count
 total=total+arrayvar(y)
 Next y
 MsgBox "Average of " & count & " scores is: " & Int(total/count)
End Sub
```

LCase

See Also      Dim                      ReDim  
                 Global                Static  
                 Option Base            UBound

## LCase Function

---

Description      Returns a copy of a string, with all uppercase letters converted to lowercase.

Syntax            **LCase**[\$](*string*\$)

| Syntax Element   | Description                                                                                                                                                                                        |
|------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| \$               | Optional. If specified the return type is <code>String</code> . If omitted the function will typically return a Variant of <code>VarType 8</code> in the function name is ( <code>String</code> ). |
| <i>string</i> \$ | A string, or an expression containing the string to use.                                                                                                                                           |

Comments        The translation is based on the country specified in the Windows Control Panel. `LCase` accepts expressions of type `String`. `LCase` accepts any type of argument and will convert the input value to a string.  
  
If the value of *string*\$ is `NULL`, a Variant of `VarType 1` (Null) is returned.

Example          This example converts a string entered by the user to lowercase.

```
Sub main
Dim userstr as String
userstr=InputBox$("Enter a string in upper and lowercase letters")
userstr=LCase$(userstr)
MsgBox "The string now is: " & userstr
End Sub
```

See Also        `UCase`

## Left Function

---

Description      Returns a string of a specified number of characters copied from the beginning of another string.

Syntax **Left**[\$](*string*\$, *length*%)

| Syntax Element   | Description                                                                                                                                                                  |
|------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| \$               | Optional. If specified, the return type is <code>String</code> . If omitted, the function will typically return a Variant of <code>VarType 8</code> ( <code>String</code> ). |
| <i>string</i> \$ | A string or an expression containing the string to copy.                                                                                                                     |
| <i>length</i> %  | The number of characters to copy.                                                                                                                                            |

Comments If *length*% is greater than the length of *string*\$, this function returns the whole string.

`Left` accepts expressions of type `String`. `Left` accepts any type of *string*\$, including numeric values, and will convert the input value to a string.

If the value of *string*\$ is `NULL`, a Variant of `VarType 1` (`Null`) is returned.

To obtain a string of a specified number of bytes, copied from the beginning of another string, use `LeftB`.

Example This example extracts a user's first name from the entire name entered.

```
Sub main
 Dim username as String
 Dim count as Integer
 Dim firstname as String
 Dim charspace
 charspace=Chr(32)
 username=InputBox("Enter your first and last name")
 count=InStr(username,charspace)
 firstname=Left(username,count)
 MsgBox "Your first name is: " &firstname
End Sub
```

See Also

|                           |                            |                      |
|---------------------------|----------------------------|----------------------|
| <code>GetField</code>     | <code>Mid</code> statement | <code>StrComp</code> |
| <code>Len</code>          | <code>Right</code>         | <code>Trim</code>    |
| <code>LTrim</code>        | <code>RTrim</code>         |                      |
| <code>Mid</code> function | <code>Str</code>           |                      |

Len

## Len Function

---

**Description** Returns the length of a string or variable.

**Syntax** Syntax A **Len**(*string\$*)

Syntax B **Len**(*varname*)

| Syntax Element  | Description                                           |
|-----------------|-------------------------------------------------------|
| <i>string\$</i> | A string or an expression that evaluates to a string. |
| <i>varname</i>  | A variable that contains a string.                    |

**Comments** If the argument is a string, the number of characters in the string is returned. If the argument is a Variant variable, **Len** returns the number of bytes required to represent its value as a string. Otherwise, the length of the built-in data type or user-defined type is returned.

If syntax B is used, and *varname* is a Variant containing a NULL, **Len** will return a Null Variant.

To return the number of bytes in a string, use **LenB**.

**Example** This example returns the length of a name entered by the user (including spaces).

```
Sub Main
 Dim username as String
 Dim Count as Integer
 username=InputBox("Enter your name")
 count=Len(username)
 MsgBox "The length of your name is: " &count
End Sub
```

**See Also** [Instr](#)

## Let Statement

---

**Description** Assigns an expression to an SQABasic variable.

**Syntax** [**Let**] *variable* = *expression*

| Syntax Element    | Description                                                 |
|-------------------|-------------------------------------------------------------|
| <i>variable</i>   | The name of a variable to assign to the <i>expression</i> . |
| <i>expression</i> | The expression to assign to the variable.                   |



- Comments**     The keyword `Let` is optional.
- The `Let` statement can be used to assign a value or expression to a variable of Numeric, String, Variant or User-Defined type. You can also use the `Let` statement to assign to an element of an array.
- When assigning a value to a numeric or string variable, standard conversion rules apply.
- `Let` differs from `Set` in that `Set` assigns a variable to an OLE object. For example:
- ▶ `Set o1 = o2` sets the object reference.
  - ▶ `Let o1 = o2` sets the value of the default member.

- Example**        This example uses the `Let` statement to assign an initial value to the variable `sum`. The sub procedure finds an average of 10 golf scores.

```
Sub main
 Dim score As Integer
 Dim x, sum
 Dim msgtext
 Let sum=0
 For x=1 to 10
 score=InputBox("Enter your last ten golf scores #" & x & ":")
 sum=sum+score
 Next x
 msgtext="Your average is: " & CInt(sum/(x-1))
 MsgBox msgtext
End Sub
```

- See Also**        `Const`  
                   `Lset`  
                   `Set`

## Like Operator

---

- Description**    Returns the value -1 (TRUE) if a string matches a pattern, 0 (FALSE) otherwise.

**Syntax**            `string$ LIKE pattern$`

| Syntax Element         | Description                                               |
|------------------------|-----------------------------------------------------------|
| <code>string\$</code>  | Any string expression.                                    |
| <code>pattern\$</code> | Any string expression to match to <code>string\$</code> . |

## Line Input

Comments *pattern*\$ can include the following special characters:

| Character:              | Matches:                                                     |
|-------------------------|--------------------------------------------------------------|
| ?                       | A single character                                           |
| *                       | A set of zero or more characters                             |
| #                       | A single digit character (0-9)                               |
| [ <i>chars</i> ]        | A single character in <i>chars</i>                           |
| [! <i>chars</i> ]       | A single character not in <i>chars</i>                       |
| [ <i>schar-echar</i> ]  | A single character in range <i>schar</i> to <i>echar</i>     |
| [! <i>schar-echar</i> ] | A single character not in range <i>schar</i> to <i>echar</i> |

Both ranges and lists can appear within a single set of square brackets. Ranges are matched according to their ANSI values. In a range, *schar* must be less than *echar*.

If either *string*\$ or *pattern*\$ is NULL then the result value is NULL.

The Like operator respects the current setting of Option Compare.

Example This example tests whether a letter is lowercase.

```
Sub main
 Dim userstr as String
 Dim revalue as Integer
 Dim retvalue as Integer
 Dim msgtext as String
 Dim pattern
 pattern="[a-z]"
 userstr=InputBox$("Enter a letter:")
 retvalue=userstr LIKE pattern
 If retvalue=-1 then
 msgtext="The letter " & userstr & " is lowercase."
 Else
 msgtext="Not a lowercase letter."
 End If
 MsgBox msgtext
End Sub
```

See Also      Expressions      Option Compare  
                 Instr              StrComp

## Line Input Statement

---

Description      Reads a line from the a sequential file or from the keyboard into a string variable.

Syntax      Syntax A    **Line Input** [#]*filename%*, *varname\$*

Syntax B    **Line Input** [*prompt\$*,] *varname\$*

| Syntax Element   | Description                                                                                    |
|------------------|------------------------------------------------------------------------------------------------|
| <i>filename%</i> | An integer expression identifying the open file to use.                                        |
| <i>prompt\$</i>  | An optional string that can be used to prompt for keyboard input; it must be a literal string. |
| <i>varname\$</i> | A string variable to contain the line read.                                                    |

Comments    If specified, the *filename%* is the number used in the `Open` statement to open the file. If *filename%* is not provided, the line is read from the keyboard.

If *prompt\$* is not provided, a prompt of a question mark ( ? ) is used.

Example      This example reads the contents of a sequential file line by line (to a carriage return) and displays the results. The second sub procedure, `CREATEFILE`, creates the file `C:\TEMP001` used by the main sub procedure.

```

Declare Sub createfile()
Sub main
 Dim msgtext as String
 Dim testscore as String
 Dim x
 Dim y
 Dim newline
 Call createfile
 Open "c:\temp001" for Input as #1
 x=1
 newline=Chr(10)
 msgtext= "The contents of c:\temp001 is: " & newline
 Do Until x=Lof(1)
 Line Input #1, testscore
 x=x+1
 y=Seek(1)
 If y>Lof(1) then
 x=Lof(1)
 Else
 Seek 1,y
 End If
 msgtext=msgtext & testscore & newline
 Loop
 MsgBox msgtext
 Close #1
 Kill "C:\TEMP001"
End Sub

Sub createfile()
 Rem Put the numbers 1-10 into a file
 Dim x as Integer
 Open "C:\TEMP001" for Output as #1
 For x=1 to 10
 Write #1, x
 Next x
 Close #1
End Sub

```

## ListBox (Statement)

See Also      Get                      InputBox  
                  Input function            Open  
                  Input statement

## ListBox Statement

---

Description    Defines a list box of choices for a dialog box.

Syntax          Syntax A    **ListBox** *x*, *y*, *dx*, *dy*, *text\$*, *.field*

                  Syntax B    **ListBox** *x*, *y*, *dx*, *dy*, *stringarray\$()*, *.field*

| Syntax Element        | Description                                                                                             |
|-----------------------|---------------------------------------------------------------------------------------------------------|
| <i>x</i> , <i>y</i>   | The upper left corner coordinates of the list box, relative to the upper left corner of the dialog box. |
| <i>dx</i> , <i>dy</i> | The width and height of the list box.                                                                   |
| <i>text\$</i>         | A string containing the selections for the list box.                                                    |
| <i>stringarray\$</i>  | An array of dynamic strings for the selections in the list box.                                         |
| <i>.field</i>         | The name of the dialog-record field that will hold a number for the choice made in the list box.        |

Comments       The *x* argument is measured in 1/4 system-font character-width units. The *y* argument is measured in 1/8 system-font character-width units. (See `Begin Dialog` for more information.)

The *text\$* argument must be defined, using a `Dim` statement, before the `Begin Dialog` statement is executed. The arguments in the *text\$* string are entered as shown in the following example:

```
dimname="listchoice"+Chr$(9)+"listchoice"+Chr$(9)+"listchoice"...
```

A number representing the selection's position in the *text\$* string is recorded in the field designated by the *.field* argument when the OK button (or any `PushButton` other than `Cancel`) is pushed. The numbers begin at 0. If no item is selected, it is -1. The *field* argument is also used by the dialog statements that act on this control.

Use the `ListBox` statement only between a `Begin Dialog` and an `End Dialog` statement.

**Example** This example defines a dialog box with a list box and two buttons.

```

Sub main
 Dim ListBox1() as String
 ReDim ListBox1(0)
 ListBox1(0)="C:\"
 Begin Dialog UserDialog 133, 66, 171, 65, "SQABasic Dialog Box"
 Text 3, 3, 34, 9, "Directory:", .Text2
 ListBox 3, 14, 83, 39, ListBox1(), .ListBox2
 OKButton 105, 6, 54, 14
 CancelButton 105, 26, 54, 14
 End Dialog
 Dim mydialog as UserDialog
 On Error Resume Next
 Dialog mydialog
 If Err=102 then
 MsgBox "Dialog box canceled."
 End If
End Sub

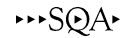
```

**See Also**

|                  |              |                |
|------------------|--------------|----------------|
| Begin/End Dialog | ComboBox     | OptionGroup    |
| Button           | Dialog       | Picture        |
| ButtonGroup      | DropComboBox | StaticComboBox |
| CancelButton     | GroupBox     | Text           |
| Caption          | OKButton     | TextBox        |
| CheckBox         | OptionButton |                |

## ListBox

User Action Command



**Description** Performs an action on a list box control.

**Syntax** **ListBox** *action%*, *recMethod\$*, *parameters\$*

| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i> | <p>One of these actions:</p> <ul style="list-style-type: none"> <li>▶ <b>Deselect.</b> Deselects the specified item from an extended Java listbox in <code>multipleMode</code>. Used only for the Java environment. <i>recMethod\$</i> must contain one of the Java recognition methods, and <i>parameters\$</i> must contain either <code>Text</code> or <code>Index</code>.</li> <li>▶ <b>ExtendSelection.</b> Selects the specified item from an extended Java listbox in <code>multipleMode</code>. Used only for the Java environment. <i>recMethod\$</i> must contain one of the Java recognition methods, and <i>parameters\$</i> must contain either <code>Text</code> or <code>Index</code>.</li> <li>▶ <b>MakeSelection.</b> Selects the specified item from a Java listbox. Used only for the Java environment. <i>recMethod\$</i> must contain one of the Java recognition methods, and <i>parameters\$</i> must contain either <code>Text</code> or <code>Index</code>.</li> </ul> |



ListBox (User Action Command)

| ▶ ▶ ▶<br>Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|-------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                         | <ul style="list-style-type: none"> <li>▶ <i>MouseClicked</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> must contain one of the following: Text, ItemData, Index, or Coords=<i>x,y</i>.</li> <li>▶ <i>MouseDown</i>. The dragging of the mouse while mouse buttons and/or shifting keys (Ctrl, Alt, Shift) are pressed. When <i>action%</i> contains a mouse-drag value, <i>parameters\$</i> must contain Coords=<i>x1,y1,x2,y2</i>. See Appendix E for a list of mouse click and drag values.</li> <li>▶ <i>ScrollAction</i>. One of these scroll actions: <ul style="list-style-type: none"> <li>ScrollPageRight      ScrollPageDown</li> <li>ScrollRight          ScrollLineDown</li> <li>ScrollPageLeft      ScrollPageUp</li> <li>ScrollLeft           ScrollLineUp</li> <li>HScrollTo            VScrollTo</li> </ul> <p>HScrollTo and VScrollTo take the required parameter Position=%.</p> <p>If Robot cannot interpret the action being applied to a scroll bar, which happens with certain custom standalone scroll bars, it records the action as a click or drag.</p> </li> </ul> |
| <i>recMethod\$</i>      | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ HTMLId=\$. The text from the ID attribute of the HTML object.</li> <li>▶ HTMLText=\$. The visible text of a Web page SELECT form element. The text is from the Value attribute of the OPTION tag.</li> <li>▶ HTMLTitle=\$. The text from the Title attribute of the HTML object.</li> <li>▶ ID=%. The object's internal Windows ID.</li> <li>▶ Index=%. The number of the object among all objects identified with the same base recognition method. Typically, Index is used after another recognition method qualifier — for example, Name=\$; Index=%.</li> <li>▶ JavaText=\$. A label that identifies the object in the user interface.</li> <li>▶ Label=\$. The text of the label object that immediately precedes the list box in the Z order of windows.</li> <li>▶ Name=\$. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be Command1.</li> </ul>                                                                                                                                                                                                                       |

▶ ▶ ▶

▶ ▶ ▶

| Syntax Element      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                     | <ul style="list-style-type: none"> <li>▶ <code>ObjectIndex=%</code>. The number of the object among all objects of the same type in the same window.</li> <li>▶ <code>State=\$</code>. An optional qualifier for any other recognition method. There are two possible values for this setting: <code>Enabled</code> and <code>Disabled</code>. The default state is the state of the current context window (as set in the most recent <code>Window SetContext</code> command), or <code>Enabled</code> if the state has not been otherwise declared.</li> <li>▶ <code>Type=\$</code>. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The <code>Type</code> qualifier uses the following form: <code>Type=\$;recMethod=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>;\;</code>).</li> <li>▶ <code>VisualText=\$</code>. An optional setting used to identify an object by its prior label. It is for user clarification only and does not affect object recognition.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <i>parameters\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>Coords=x,y</code>. If <i>action%</i> is a mouse click, specifies the coordinates of the click, relative to the top left of the object. Robot uses this parameter only if the item contents or index cannot be retrieved — for example, if the list box is empty or disabled.</li> <li>▶ <code>Coords=x1,x2,y1,y2</code>. If <i>action%</i> is a mouse drag, specifies the coordinates, where <i>x1,y1</i> are the starting coordinates of the drag, and <i>x2,y2</i> are the ending coordinates. The coordinates are relative to the top left of the object.</li> <li>▶ <code>Index=%</code>. If <i>action%</i> is a select or deselect action, identifies the index of an item in the list.</li> <li>▶ <code>ItemData=&amp;</code>. If <i>action%</i> is a mouse click, identifies the internal value, or <code>ItemData</code>, associated with an item in the list. All items in a list have an associated value. The uniqueness and significance of this value is entirely up to the application. Robot uses this parameter only if the list box item's text cannot be retrieved (for example, if it is an Owner Drawn list box), and if the Identify List Selections By recording option is set to <code>Contents</code>.</li> <li>▶ <code>Position=%</code>. If <i>action%</i> is a <code>VScrollTo</code> or <code>HScrollTo</code>, specifies the scroll bar value of the new scrolled-to position. Every scroll bar has an internal range, and this value is specific to that range.</li> <li>▶ <code>Text=\$</code>. If <i>action%</i> is a select or deselect action, identifies the text of an item in the list.</li> </ul> |

ListBoxVP

Comments None.

Example This example clicks the item identified by the text `Epson` on `LPT1`: in the first list box control in the window (`ObjectIndex=1`).

```
ListBox Click, "ObjectIndex=1", "Text=Epson on LPT1:"
```

This example clicks the item identified by the text `Option 1` in the list box named `SelectList1`. The list box is located within the Web page frame named `Main`.

```
ListBox Click,
"Type=HTMLFrame;HTMLId=Main;\;Type=ListBox; Name=SelectList1",
"Text=Option 1"
```

See Also      `ComboBox`              `ComboListBox`  
                 `ComboEditBox`      `EditBox`

## ListBoxVP

Verification Point Command

»»SQA»

Description Establishes a verification point for a list box control.

Syntax *Result* = **ListBoxVP** (*action%*, *recMethod\$*, *parameters\$*)

| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i> | <p>The type of verification to perform. Valid values:</p> <ul style="list-style-type: none"><li>▶ <b>Compare</b>. Captures the entire contents of the list box into a grid and compares it to a recorded baseline. <i>parameters\$ VP</i> is required; <i>ExpectedResult</i> and <i>Wait</i> are optional.</li><li>▶ <b>CompareData</b>. Captures the contents or HTML text of the object and compares it to a recorded baseline. <i>parameters\$ VP</i> is required; <i>ExpectedResult</i> and <i>Wait</i> are optional.</li><li>▶ <b>CompareNumeric</b>. Captures the numeric value of the text of the object and compares it to the value of <i>parameters\$ Value</i> or <i>Range</i>. <i>parameters\$ VP</i> and either <i>Value</i> or <i>Range</i> are required; <i>ExpectedResult</i> and <i>Wait</i> are optional.</li><li>▶ <b>CompareProperties</b>. Captures object properties information for the object and compares it to a recorded baseline. <i>parameters\$ VP</i> is required; <i>ExpectedResult</i> and <i>Wait</i> are optional.</li></ul> |

▶ ▶ ▶



▶ ▶ ▶

| Syntax Element      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                     | <ul style="list-style-type: none"> <li>▶ <code>CompareText</code>. Captures the text of the object and compares it to a recorded baseline. <i>parameters\$ VP</i> and <code>Type</code> are required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li> <li>▶ <code>VerifyIsBlank</code>. Checks that the object has no text. <i>parameters\$ VP</i> is required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <i>recMethod\$</i>  | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>HTMLId=\$</code>. The text from the ID attribute of the HTML object.</li> <li>▶ <code>HTMLText=\$</code>. The visible text of a Web page SELECT form element. The text is from the Value attribute of the OPTION tag.</li> <li>▶ <code>HTMLTitle=\$</code>. The text from the Title attribute of the HTML object.</li> <li>▶ <code>ID=%</code>. The object's internal Windows ID.</li> <li>▶ <code>Index=%</code>. The number of the object among all objects identified with the same base recognition method. Typically, <code>Index</code> is used after another recognition method qualifier — for example, <code>Name=\$; Index=%</code>.</li> <li>▶ <code>JavaText=\$</code>. A label that identifies the object in the user interface.</li> <li>▶ <code>Label=\$</code>. The text of the label object that immediately precedes the list box in the Z order of windows.</li> <li>▶ <code>Name=\$</code>. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be <code>Command1</code>.</li> <li>▶ <code>ObjectIndex=%</code>. The number of the object among all objects of the same type in the same window.</li> <li>▶ <code>Type=\$</code>. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The <code>Type</code> qualifier uses the following form: <code>Type=\$; recMethod=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>;\;</code>).</li> </ul> |
| <i>parameters\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>ExpectedResult=%</code>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result).</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

▶ ▶ ▶



| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                | <p><code>ExpectedResult</code> takes either of these values:</p> <ul style="list-style-type: none"> <li>- <code>PASS</code>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail.</li> <li>- <code>FAIL</code>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail.</li> </ul> <ul style="list-style-type: none"> <li>▶ <code>Range=&amp;, &amp;</code>. Used with the action <code>CompareNumeric</code> when a numeric range comparison is being performed, as in <code>Range=2, 12</code> (test for numbers in this range). The values are inclusive.</li> <li>▶ <code>Type=\$</code>. Specifies the verification method to use for <code>CompareText</code> actions. The possible values are: <code>CaseSensitive</code>, <code>CaseInsensitive</code>, <code>FindSubStr</code>, <code>FindSubStrI</code> (case insensitive), and <code>UserDefined</code>. See <i>Comments</i> for more information. If <code>UserDefined</code> is specified, two additional parameters are required: <ul style="list-style-type: none"> <li>- <code>DLL=\$</code>. The full path and file name of the library that contains the function</li> <li>- <code>Function=\$</code>. The name of the custom function to use in comparing the text</li> </ul> </li> <li>▶ <code>Value=&amp;</code>. Used with the action <code>CompareNumeric</code> when a numeric equivalence comparison is being performed, as in <code>Value=25</code> (test against the value 25).</li> <li>▶ <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points.</li> <li>▶ <code>Wait=%, %</code>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10, 40</code> (retry the test every 10 seconds, but time out the test after 40 seconds).</li> </ul> |

**Comments** This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

With the `Type=$` parameter, `CaseSensitive` and `CaseInsensitive` require a full match between the current baseline text and the text captured during playback. With `FindSubStr` and `FindSubStrI`, the current baseline can be a substring of the text captured during playback. The substring can appear anywhere in the playback text. To modify the current baseline text, double-click the verification point name in the Robot Asset pane (to the left of the script).

**Example** This example captures the properties of the list box labeled `Files:` and compares them to the recorded baseline in verification point `FILELIST`.

```
Result = ListViewVP(CompareProperties,"Label=Files:","VP=FILELIST")
```

This example captures the data from the list box identified by the name `SelectList1`. The list is located within the Web page frame named `Main`. `ListViewVP` compares the data to the recorded baseline in verification point `WebList1`.

```
Result = ListViewVP (CompareData,
 "Type=HTMLFrame;HTMLID=main;\;Type=ListBox;Name=SelectList1",
 "VP=WebList1")
```

**See Also** [ComboBoxVP](#) [EditBoxVP](#)  
[ComboEditBoxVP](#)

## ListView

User Action Command



**Description** Performs an action on a list view control.

**Syntax** `ListView action%, recMethod$, parameters$`

| Syntax Element       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>action%</code> | <p>One of these actions:</p> <ul style="list-style-type: none"> <li>▶ <i>MouseClicked</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <code>action%</code> contains a mouse-click value, <code>parameters\$</code> must contain <code>Coords=x,y</code>.</li> <li>▶ <i>MouseDown</i>. The dragging of the mouse while mouse buttons and/or shifting keys (Ctrl, Alt, Shift) are pressed. When <code>action%</code> contains a mouse-drag value, <code>parameters\$</code> must contain <code>Coords=x1,y1,x2,y2</code>. See Appendix E for a list of mouse click and drag values.</li> <li>▶ <i>ScrollAction</i>. One of these scroll actions: <ul style="list-style-type: none"> <li>ScrollPageRight      ScrollPageDown</li> <li>ScrollRight          ScrollLineDown</li> <li>ScrollPageLeft      ScrollPageUp</li> <li>ScrollLeft           ScrollLineUp</li> <li>HScrollTo            VScrollTo</li> </ul> <p>HScrollTo and VScrollTo take the required parameter <code>Position=%</code>.</p> <p>If Robot cannot interpret the action being applied to a scroll bar, which happens with certain custom standalone scroll bars, it records the action as a click or drag.</p> </li> </ul> |



ListView

▶ ▶ ▶

| Syntax Element       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>recMethod</i> \$  | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ ID=%. The object's internal Windows ID.</li> <li>▶ ItemIndex=%. The index of the list view item acted upon. Used only after one of these parent values: ID=%, ObjectIndex=%, Name=\$, Text=\$. Parent/child values are separated by a backslash and semicolons (; \ ;).</li> <li>▶ ItemText=\$. The text of the list view item acted upon. Used only after one of these parent values: ID=%, ObjectIndex=%, Name=\$, Text=\$. Parent/child values are separated by a backslash and semicolons (; \ ;).</li> <li>▶ Name=\$. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be Command1.</li> <li>▶ ObjectIndex=%. The number of the object among all objects of the same type in the same window.</li> <li>▶ State=\$. An optional qualifier for any other recognition method. There are two possible values for this setting: Enabled and Disabled. The default state is the state of the current context window (as set in the most recent Window SetContext command), or Enabled if the state has not been otherwise declared.</li> <li>▶ Text=\$. The text displayed on the object.</li> <li>▶ VisualText=\$. An optional setting used to identify an object by its prior label. It is for user clarification only and does not affect object recognition.</li> </ul> |
| <i>parameters</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ Coords=<i>x,y</i>. If <i>action%</i> is a mouse click, specifies the <i>x,y</i> coordinates of the click, relative to the top left of the object or the item.</li> <li>▶ Coords=<i>x1,y1,x2,y2</i>. If <i>action%</i> is a mouse drag, specifies the coordinates, where <i>x1,y1</i> are the starting coordinates of the drag, and <i>x2,y2</i> are the ending coordinates. The coordinates are relative to the top left of the object or the item.</li> <li>▶ Position=%. If <i>action%</i> is VScrollTo or HScrollTo, specifies the scroll bar value of the new scrolled-to position in the scroll box. Every scroll bar has an internal range, and this value is specific to that range.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |

- Comments    None.
- Example      This example clicks the item identified by the text *System* at *x,y* coordinates of 50,25 in the first list view control in the window (ObjectIndex=1).  
                  `ListView Click, "ObjectIndex=1;\;ItemText=System", "Coords=50,25"`
- See Also      ListViewVP

## ListViewVP

Verification Point Command



Description    Establishes a verification point for a list view control.

Syntax          `Result = ListViewVP (action%, recMethod$, parameters$)`

| Syntax Element     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i>     | <p>The type of verification to perform. Valid values:</p> <ul style="list-style-type: none"> <li>▶ CompareData. Captures the data of the object and compares it to a recorded baseline. <i>parameters\$ VP</i> is required; ExpectedResult and Wait are optional.</li> <li>▶ CompareNumeric. Captures the numeric value of the text of the object and compares it to the value of <i>parameters\$ Value or Range</i>. <i>parameters\$ VP</i> and either Value or Range are required; ExpectedResult and Wait are optional.</li> <li>▶ CompareProperties. Captures object properties information for the object and compares it to a recorded baseline. <i>parameters\$ VP</i> is required; ExpectedResult and Wait are optional.</li> <li>▶ CompareText. Captures the text of the object and compares it to a recorded baseline. <i>parameters\$ VP</i> and Type are required; ExpectedResult and Wait are optional.</li> </ul> |
| <i>recMethod\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ ID=%. The object's internal Windows ID.</li> <li>▶ ItemIndex=%. The index of the list view item acted upon. Used only after one of these parent values: ID=%, ObjectIndex=%, Name=\$, Text=\$. Parent/child values are separated by a backslash and semicolons (; \;).</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |



▶ ▶ ▶

| Syntax Element      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                     | <ul style="list-style-type: none"> <li>▶ <code>ItemText=\$</code>. The text of the list view item acted upon. Used only after one of these parent values: <code>ID=%</code>, <code>ObjectIndex=%</code>, <code>Name=\$</code>, <code>Text=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>;\</code>).</li> <li>▶ <code>Name=\$</code>. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be <code>Command1</code>.</li> <li>▶ <code>ObjectIndex=%</code>. The number of the object among all objects of the same type in the same window.</li> <li>▶ <code>Text=\$</code>. The text displayed on the object.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <i>parameters\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>ExpectedResult=%</code>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values:             <ul style="list-style-type: none"> <li>– <code>PASS</code>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail.</li> <li>– <code>FAIL</code>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail.</li> </ul> </li> <li>▶ <code>Range=&amp; , &amp;</code>. Used with the action <code>CompareNumeric</code> when a numeric range comparison is being performed, as in <code>Range=2 , 12</code> (test for numbers in this range). The values are inclusive.</li> <li>▶ <code>Type=\$</code>. Specifies the verification method to use for <code>CompareText</code> actions. The possible values are: <code>CaseSensitive</code>, <code>CaseInsensitive</code>, <code>FindSubStr</code>, <code>FindSubStrI</code> (case insensitive), and <code>UserDefined</code>. See <i>Comments</i> for more information. If <code>UserDefined</code> is specified, two additional parameters are required:             <ul style="list-style-type: none"> <li>– <code>DLL=\$</code>. The full path and file name of the library that contains the function</li> <li>– <code>Function=\$</code>. The name of the custom function to use in comparing the text</li> </ul> </li> <li>▶ <code>Value=&amp;</code>. Used with the action <code>CompareNumeric</code> when a numeric equivalence comparison is being performed, as in <code>Value=25</code> (test against the value 25).</li> </ul> |

▶ ▶ ▶

▶ ▶ ▶

| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                | <ul style="list-style-type: none"> <li>▶ <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points.</li> <li>▶ <code>Wait=%, %</code>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10, 40</code> (retry the test every 10 seconds, but time out the test after 40 seconds).</li> </ul> |

**Comments** This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

With the `Type=$` parameter, `CaseSensitive` and `CaseInsensitive` require a full match between the current baseline text and the text captured during playback. With `FindSubStr` and `FindSubStrI`, the current baseline can be a substring of the text captured during playback. The substring can appear anywhere in the playback text. To modify the current baseline text, double-click the verification point name in the Robot Asset pane (to the left of the script).

**Example** This example captures the properties of the first list view control in the window (`ObjectIndex=1`) and compares them to the recorded baseline in verification point TEST1A.

```
Result = ListViewVP (CompareProperties, "ObjectIndex=1", "VP=TEST1A")
```

**See Also** `ListView`

## Loc Function

**Description** Returns the current offset within an open file.

**Syntax** `Loc (filenumber%)`

| Syntax Element           | Description                                               |
|--------------------------|-----------------------------------------------------------|
| <code>filenumber%</code> | An integer expression identifying the open file to query. |

**Comments** The `filenumber%` is the number used in the `Open` statement of the file.

For files opened in `Random` mode, `Loc` returns the number of the last record read or written. For files opened in `Append`, `Input`, or `Output` mode, `Loc` returns the current byte offset divided by 128. For files opened in `Binary` mode, `Loc` returns the offset of the last byte read or written.

## Lock

### Example

This example creates a file of account numbers as entered by the user. When the user finishes, the example displays the offset in the file of the last entry made.

```
Sub main
 Dim filepos as Integer
 Dim acctno() as Integer
 Dim x as Integer
 x=0
 Open "c:\TEMP001" for Random as #1
 Do
 x=x+1
 Redim Preserve acctno(x)
 acctno(x)=InputBox("Enter account #" & x & " or 0 to end:")
 If acctno(x)=0 then
 Exit Do
 End If
 Put #1,, acctno(x)
 Loop
 filepos=Loc(1)
 Close #1
 MsgBox "The offset is: " & filepos
 Kill "C:\TEMP001"
End Sub
```

### See Also

Eof  
Lof  
Open

## Lock Statement

---

**Description** Keeps other processes from accessing an open file.

**Syntax** **Lock** [#]*filenumber%* [, [*start&*] [*To end&*]]

| Syntax Element     | Description                                               |
|--------------------|-----------------------------------------------------------|
| <i>filenumber%</i> | An integer expression identifying the open file.          |
| <i>start&amp;</i>  | Number of the first record or byte offset to lock/unlock. |
| <i>end&amp;</i>    | Number of the last record or byte offset to lock/unlock.  |

**Comments** The *filenumber%* is the number used in the `Open` statement of the file.

For `Binary` mode, *start&*, and *end&* are byte offsets. For `Random` mode, *start&*, and *end&* are record numbers. If *start&* is specified without *end&*, only the record or byte at *start&* is locked. If *To end&* is specified without *start&*, all records or bytes from record number or offset 1 to *end&* are locked.

For `Input`, `Output` and `Append` modes, *start&*, and *end&* are ignored and the whole file is locked.



`Lock` and `Unlock` always occur in pairs with identical parameters. All locks on open files must be removed before closing the file or unpredictable results occur.

### Example

This example locks a file that is shared by others on a network, if the file is already in use. The second sub procedure, `CREATEFILE`, creates the file used by the main sub procedure.

```

Declare Sub createfile
Sub main
 Dim btngrp, icongrp
 Dim defgrp
 Dim answer
 Dim noaccess as Integer
 Dim msgabort
 Dim msgstop as Integer
 Dim acctname as String
 noaccess=70
 msgstop=16
 Call createfile
 On Error Resume Next
 btngrp=1
 icongrp=64
 defgrp=0
 answer=MsgBox("Open the account file?" & Chr(10),
 btngrp+icongrp+defgrp)
 If answer=1 then
 Open "C:\TEMP001" for Input as #1
 If Err=noaccess then
 msgabort=MsgBox("File Locked",msgstop,"Aborted")
 Else
 Lock #1
 Line Input #1, acctname
 MsgBox "The first account name is: " & acctname
 Unlock #1
 End If
 Close #1
 End If
 Kill "C:\TEMP001"
End Sub

Sub createfile()
 Rem Put the letters A-J into the file
 Dim x as Integer
 Open "C:\TEMP001" for Output as #1
 For x=1 to 10
 Write #1, Chr(x+64)
 Next x
 Close #1
End Sub

```

### See Also

`Open`  
`Unlock`

Lof

## Lof Function

---

**Description** Returns the length in bytes of an open file.

**Syntax** **Lof**(*filenumber%*)

| Syntax Element     | Description                                      |
|--------------------|--------------------------------------------------|
| <i>filenumber%</i> | An integer expression identifying the open file. |

**Comments** The *filenumber%* is the number used in the `Open` statement of the file.

**Example** This example opens a file and prints its contents to the screen.

```
Sub main
 Dim fname
 Dim fchar()
 Dim x as Integer
 Dim msgtext
 Dim newline
 newline=Chr(10)
 fname=InputBox("Enter a filename to print:")
 On Error Resume Next
 Open fname for Input as #1
 If Err<>0 then
 MsgBox "Error loading file. Re-run program."
 Exit Sub
 End If
 msgtext="The contents of " & fname & " is: " &
 newline &newline
 Redim fchar(Lof(1))
 For x=1 to Lof(1)
 fchar(x)=Input(1,#1)
 msgtext=msgtext & fchar(x)
 Next x
 MsgBox msgtext
 Close #1
End Sub
```

**See Also** Eof            Loc  
FileLen        Open

## Log Function

---

**Description** Returns the natural logarithm of a number.

**Syntax** **Log** (*number*)

| Syntax Element | Description                   |
|----------------|-------------------------------|
| <i>number</i>  | Any valid numeric expression. |

**Comments** The return value is single-precision for an integer, currency or single-precision numeric expression, double precision for a long, Variant or double-precision numeric expression.

**Example** This example uses the `Log` function to determine which number is larger:  $999^{1000}$  (999 to the 1000 power) or  $1000^{999}$  (1000 to the 999 power). Note that you cannot use the exponent (^) operator for numbers this large.

```
Sub main
 Dim a as Integer
 Dim b as Integer
 Dim x
 Dim y
 x=999
 y=1000
 a=y*(Log(x))
 b=x*(Log(y))
 If a>b then
 MsgBox "999^1000 is greater than 1000^999"
 Else
 MsgBox "1000^999 is greater than 999^1000"
 End If
End Sub
```

**See Also**

|                  |                  |
|------------------|------------------|
| <code>Exp</code> | <code>Rnd</code> |
| <code>Fix</code> | <code>Sgn</code> |
| <code>Int</code> | <code>Sqr</code> |

Lset

## Lset Statement

---

**Description** Copies one string to another, or assigns a user-defined type variable to another.

**Syntax** Syntax A **Lset** *string\$* = *string-expression*

Syntax B **Lset** *variable1* = *variable2*

| Syntax Element           | Description                                                         |
|--------------------------|---------------------------------------------------------------------|
| <i>string\$</i>          | A string or string expression to contain the copied characters.     |
| <i>string-expression</i> | An expression containing the string to copy.                        |
| <i>variable1</i>         | A variable with a user-defined type to contain the copied variable. |
| <i>variable2</i>         | A variable with a user-defined type to copy.                        |

**Comments** If *string\$* is shorter than *string-expression*, **Lset** copies the leftmost character of *string-expression* into *string\$*. The number of characters copied is equal to the length of *string\$*.

If *string\$* is longer than *string-expression*, all characters of *string-expression* are copied into *string\$*, filling it from left to right. All leftover characters of *string\$* are replaced with spaces.

In Syntax B, the number of characters copied is equal to the length of the shorter of *variable1* and *variable2*.

**Lset** cannot be used to assign variables of different user-defined types if either contains a **Variant** or a variable-length string.

**Example** This example puts a user's last name into the variable **LASTNAME**. If the name is longer than the size of **LASTNAME**, then the user's name is truncated. If you have a long last name and you get lots of junk mail, you've probably seen how this works already.

```
Sub main
 Dim msgtext, lastname as String
 Dim strlast as String*8
 lastname=InputBox("Enter your last name")
 Lset strlast=lastname
 msgtext="Your last name is: " & strlast
 MsgBox msgtext
End Sub
```

**See Also** **Rset**

## LTrim

### Function

---

**Description** Returns a copy of a string with all leading space characters removed.

**Syntax** **LTrim**[\$](*expression*)

| Syntax Element    | Description                                                                                                                                                |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| \$                | Optional. If specified, the return type is <code>String</code> . If omitted, the function typically returns a Variant of <code>VarType 8 (String)</code> . |
| <i>expression</i> | The expression to trim. The expression can be a string, or it can be a numeric data type which Robot passes to the command as a string.                    |

**Comments** If the value of *string\$* is NULL, a Variant of `VarType 1 (Null)` is returned.

**Example** This example trims the leading spaces from a string padded with spaces on the left.

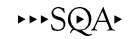
```
Sub main
 Dim userInput as String
 Dim numsize
 Dim str1 as String*50
 Dim strsize
 strsize=50
 userInput=InputBox("Enter a string of characters:")
 numsize=Len(userInput)
 str1=Space(strsize-numsize) & userInput
 ' Str1 has a variable number of leading spaces.
 MsgBox "The string is: " &str1
 str1=LTrim$(str1)
 ' Str1 now has no leading spaces.
 MsgBox "The string now has no leading spaces: " & str1
End Sub
```

**See Also**

|               |       |
|---------------|-------|
| GetField      | Right |
| Left          | RTrim |
| Mid function  | Trim  |
| Mid statement |       |

## MenuIDSelect

User Action Command



**Description** Performs a menu selection based on the internal ID of the menu item.

**Syntax** **MenuIDSelect** *MenuID&*

| Syntax Element     | Description                     |
|--------------------|---------------------------------|
| <i>MenuID&amp;</i> | The internal ID of a menu item. |

**Comments** This command is necessary for making selections from menu items that do not contain text, such as owner drawn or bitmap menus.

**Example** This example selects the menu item identified by the internal ID 1145 from the current context window.

```
MenuIDSelect 1145
```

**See Also** MenuSelect                      SysMenuIDSelect  
 PopupMenuIDSelect                  SysMenuSelect  
 PopupMenuSelect

## MenuSelect

User Action Command



**Description** Selects a menu item through one or more mouse clicks.

**Syntax** **MenuSelect** *menuPath\$*

| Syntax Element    | Description                                                                                                                                                                                                                                                                                                                                                             |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>menuPath\$</i> | A sequential list of the menu, any sub-menus, and the target menu item that a user clicks. Each is separated by a pointer (->).<br><br>If you are specifying an item by position or by ID rather than by name, <i>menuPath</i> must begin with Menu=. For example, Menu=File->pos(3) selects the third item in the File menu. See <i>Comments</i> for more information. |

Comments During recording, Robot identifies menu item selections by item name. Each name represents a mouse click. For example, Robot might record a command to open a file as follows:

```
MenuSelect "File->Open..." ' User clicks File, then Open
```

During manual scripting, you can reference a menu item selection in any of the following ways. All of the following examples select the About Rational Administrator... item of the Rational Administrator Help menu:

- ▶ Through the menu item name:

```
MenuSelect "Help->About Rational Administrator..."
```

- ▶ Through the position of the menu item on the menu:

```
MenuSelect "Menu=Help->pos(4)"
```

The first item in a menu is position 1, not 0. Also, ignore menu item separators when counting the position of an item in a menu.

- ▶ Through the menu item ID:

```
MenuSelect "Menu=Help->id(32884)"
```

You can use any of the above methods to represent both intermediate menu items and the target menu item.

When using `MenuSelect` to select a menu item, you must reference the top-level menu and every lower-level menu up to and including the menu where the target item is located. However, you can select a menu item directly by its item ID, without specifying any menu or sub-menu, by calling `MenuIDSelect`.

During manual scripting, you can select a menu item through a series of `InputKeys` commands, or through a combination of `MenuSelect` and `InputKeys` commands. This feature lets you play back a menu item selection entirely through keystrokes, or through a combination of keystrokes and mouse clicks, rather than through mouse clicks alone. For example, the following commands select the menu item Computer from the Microsoft Explorer's Tools menu and Find sub-menu:

```
Window SetContext, "Caption={Exploring*}", ""
MenuSelect "Tools" ' MenuSelect "menu=pos(4)" also works
InputKeys "f"
InputKeys "c"
```

If a menu is selected, you can clear it by calling `MenuSelect ""`.

Example This example selects the sub-menu item Change System Settings... from the top-level Options menu of the current context window.

```
MenuSelect "Options->Change System Settings..."
```

See Also `MenuIDSelect` `SysMenuIDSelect`  
`PopupMenuIDSelect` `SysMenuSelect`  
`PopupMenuSelect`

Mid (Function)

## Mid Function

---

**Description** Returns a portion of a string, starting at a specified character position.

**Syntax** **Mid**[\$](*string*\$, *start*% [, *length*%])

| Syntax Element   | Description                                                                                                                                                |
|------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| \$               | Optional. If specified, the return type is <code>String</code> . If omitted, the function typically returns a Variant of <code>VarType 8 (String)</code> . |
| <i>string</i> \$ | A string or expression that contains the string to retrieve.                                                                                               |
| <i>start</i> %   | The starting position in <i>string</i> \$ where the string to retrieve begins.                                                                             |
| <i>length</i> %  | An optional argument specifying the number of characters to retrieve.                                                                                      |

**Comments** Upon successful execution, `Mid` returns the string retrieved from *string*\$.  
`Mid` accepts any type of *string*\$, including numeric values, and will convert the input value to a string.

If the *length*% argument is omitted, or if *string*\$ is smaller than *length*%, `Mid` returns all characters from *start*% through the end of *string*\$. If *start*% is larger than *string*\$, `Mid` returns a null string ("").

The index of the first character in a string is 1.

If the value of *string*\$ is `Null`, a Variant of `VarType 1 (Null)` is returned.  
`Mid`\$ requires the string argument to be of type `string` or variant. `Mid` allows the string argument to be of any data type.

To modify a portion of a string value, see `Mid Statement`.

To return a specified number of bytes from a string, use `MidB`. With `MidB`, *start*% specifies a byte position, and *length*% specifies a number of bytes.



**Example** This example uses the `Mid` function to find the last name in a string.

```
Sub main
 Dim username as String
 Dim position as Integer
 username=InputBox("Enter your full name:")
 Do
 position=InStr(username," ")
 If position=0 then
 Exit Do
 End If
 position=position+1
 username=Mid(username,position)
 Loop
 MsgBox "Your last name is: " & username
End Sub
```

**See Also**

|                       |                            |                    |
|-----------------------|----------------------------|--------------------|
| <code>GetField</code> | <code>Len</code>           | <code>Right</code> |
| <code>LCase</code>    | <code>LTrim</code>         | <code>RTrim</code> |
| <code>Left</code>     | <code>Mid statement</code> | <code>Trim</code>  |

## Mid Statement

---

**Description** Replaces part (or all) of one string with another, starting at a specified location.

**Syntax** `Mid (stringvar$, start% [, length%]) = string$`

| Syntax Element     | Description                                      |
|--------------------|--------------------------------------------------|
| <i>stringvar\$</i> | The string to change.                            |
| <i>start%</i>      | The position where character replacement begins. |
| <i>length%</i>     | The number of characters to replace.             |
| <i>string\$</i>    | The string to place into <i>stringvar\$</i> .    |

**Comments** If the *length%* argument is omitted, or if there are fewer characters in *string\$* than specified in *length%*, then `Mid` replaces all the characters from the *start%* to the end of the *string\$*. If *start%* is larger than the number of characters in the indicated *stringvar\$*, then `Mid` appends *string\$* to *stringvar\$*.

If *length%* is greater than the length of *string\$*, then *length%* is set to the length of *string\$*. If *start%* is greater than the number of characters in *stringvar\$*, an illegal function call error will occur at runtime. If *length%* plus *start%* is greater than the length of *stringvar\$*, then only the characters up to the end of *stringvar\$* are replaced.

`Mid` never changes the number of characters in *stringvar\$*.

## Minute

The index of the first character in a string is 1.

To replace a specified number of bytes in a string with those from another string, use `MidB`. With `MidB`, *start%* specifies a byte position, and *length%* specifies a number of bytes.

**Example** This example uses the `Mid` statement to replace the last name in a user-entered string to asterisks(\*).

```
Sub main
 Dim username as String
 Dim position as Integer
 Dim count as Integer
 Dim uname as String
 Dim replacement as String
 Dim x as Integer
 username=InputBox("Enter your full name:")
 uname=username
 replacement="*"
 Do
 position=InStr(username," ")
 If position=0 then
 Exit Do
 End If
 username=Mid(username,position+1)
 count=count+position
 Loop
 For x=1 to Len(username)
 count=count+1
 Mid(uname,count)=replacement
 Next x
 MsgBox "Your name now is: " & uname
End Sub
```

**See Also**

|                       |                           |
|-----------------------|---------------------------|
| <code>GetField</code> | <code>Mid</code> function |
| <code>Left</code>     | <code>Right</code>        |
| <code>Len</code>      | <code>RTrim</code>        |
| <code>LTrim</code>    | <code>Trim</code>         |

## Minute Function

---

**Description** Returns an integer for the minute component (0-59) of a date-time value.

**Syntax** `Minute(time)`

| Syntax Element | Description                                            |
|----------------|--------------------------------------------------------|
| <i>time</i>    | Any expression that can evaluate to a date-time value. |

**Comments**     `Minute` accepts any type of *time*, including strings, and will attempt to convert the input value to a date value.

The return value is a Variant of VarType 2 (Integer). If the value of *time* is null, a Variant of VarType 1 (null) is returned.

**Example**        This example extracts just the time (hour, minute, and second) from a file's last modification date and time.

```
Sub main
 Dim filename as String
 Dim ftime
 Dim hr, min
 Dim sec
 Dim msgtext as String
i: msgtext="Enter a filename:"
 filename=InputBox(msgtext)
 If filename="" then
 Exit Sub
 End If
 On Error Resume Next
 ftime=FileDateTime(filename)
 If Err<>0 then
 MsgBox "Error in file name. Try again."
 Goto i:
 End If
 hr=Hour(ftime)
 min=Minute(ftime)
 sec=Second(ftime)
 MsgBox "The file's time is: " & hr & ":" & min & ":" & sec
End Sub
```

**See Also**

|            |                |           |
|------------|----------------|-----------|
| DateSerial | Now            | TimeValue |
| DateValue  | Second         | Weekday   |
| Day        | Time function  | Year      |
| Hour       | Time statement |           |
| Month      | TimeSerial     |           |

## MkDir

### Statement

---

**Description**    Creates a new directory.

**Syntax**         **MkDir** *path*\$

| Syntax Element | Description                                                          |
|----------------|----------------------------------------------------------------------|
| <i>path</i> \$ | A string expression identifying the new default directory to create. |

## ModuleVP

Comments    The syntax for *path\$* is:

```
[drive:][\]directory[\directory]
```

The *drive* argument is optional. If *drive* is omitted, `MkDir` makes a new directory on the current drive. The *directory* argument is any directory name.

Example     This example makes a new temporary directory in `C:\` and then deletes it.

```
Sub main
 Dim path as String
 Dim C as String
 On Error Resume Next
 path=CurDir(C)
 If path<>"C:\" then
 ChDir "C:\"
 End If
 MkDir "C:\TEMP01"
 If Err=75 then
 MsgBox "Directory already exists"
 Else
 MsgBox "Directory C:\TEMP01 created"
 MsgBox "Now removing directory"
 Rmdir "C:\TEMP01"
 End If
End Sub
```

See Also

|                      |                    |
|----------------------|--------------------|
| <code>ChDir</code>   | <code>Dir</code>   |
| <code>ChDrive</code> | <code>Rmdir</code> |
| <code>CurDir</code>  |                    |

## ModuleVP

Verification Point Command

»»SQAS»

Description    Verifies whether a specified module is in memory during playback.

Syntax        *Result* = **ModuleVP** (*action%*, *recMethod\$*, *parameters\$*)

| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i> | <p>The type of verification to perform. Valid values:</p> <ul style="list-style-type: none"><li>▶ <b>Exists</b>. Checks whether the specified module is in memory. <i>parameters\$</i> VP is required; <i>ExpectedResult</i> and <i>Wait</i> are optional.</li><li>▶ <b>DoesNotExist</b>. Checks whether the specified module is not in memory. <i>parameters\$</i> VP is required; <i>ExpectedResult</i> and <i>Wait</i> are optional.</li><li>▶ <b>Note</b>: This action cannot be accessed during recording. It must be inserted manually.</li></ul> |

▶ ▶ ▶

▶ ▶ ▶

| Syntax Element       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>recMethod</i> \$  | <p>Valid value:</p> <ul style="list-style-type: none"> <li>▶ <i>Name</i>=\$. The name of the module to be verified. The name must include the three-character extension and may optionally include a fully qualified path.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <i>parameters</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <i>ExpectedResult</i>=% . Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> <li>– <i>PASS</i>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail.</li> <li>– <i>FAIL</i>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail.</li> </ul> </li> <li>▶ <i>VP</i>=\$. The verification point ID. IDs must be unique within a script. Required for all verification points.</li> <li>▶ <i>Wait</i>=% , % . A Wait State that specifies the verification point's Retry and Timeout values, as in <i>Wait</i>=10 , 40 (retry the test every 10 seconds, and timeout after 40 seconds).</li> </ul> |

**Comments** Within Microsoft Windows, modules are defined as programs (.EXE), libraries (.DLL or other), device drivers (.SYS or .DRV), and display fonts (.FON).

Verification points that check for a module's existence are not stored in the repository and do not appear in Robot's Asset pane.

**Example** This example verifies the existence in memory of the module named `USER.EXE`.

```
Result = ModuleVP (Exists, "Name=USER.EXE", "VP=MOD01")
```

**See Also** `FileVP`

Month

## Month Function

---

**Description** Returns an integer for the month component (1-12) of a date-time value.

**Syntax** **Month**(*date*)

| Syntax Element | Description                                         |
|----------------|-----------------------------------------------------|
| <i>date</i>    | Any expression that evaluates to a date-time value. |

**Comments** This function accepts any type of *date*, including strings, and will attempt to convert the input value to a date value.

The return value is a Variant of VarType 2 (integer). If the value of *date* is null, a Variant of VarType 1 (null) is returned.

**Example** This example finds the month (1-12) and day (1-31) values for this Thursday.

```
Sub main
 Dim x, today
 Dim msgtext
 Today=DateValue(Now)
 Let x=0
 Do While Weekday(Today+x)<> 5
 x=x+1
 Loop
 msgtext="This Thursday is: " & Month(Today+x)&"/"&Day(Today+x)
 MsgBox msgtext
End Sub
```

**See Also**

|                |            |           |
|----------------|------------|-----------|
| Date function  | Hour       | TimeValue |
| Date statement | Minute     | Weekday   |
| DateSerial     | Now        | Year      |
| DateValue      | Second     |           |
| Day            | TimeSerial |           |

## MsgBox Function

---

**Description** Displays a message box and returns a value (1-7) indicating which button the user selected.

Syntax **MsgBox**(*prompt*\$,[*buttons*%][, *title*\$])

| Syntax Element   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>prompt</i> \$ | The text to display in a dialog box.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <i>buttons</i> % | <p>An integer value for the buttons, the icon, and the default button choice to display in a dialog box. <i>buttons</i>% is the sum of three values, one from each of the following groups:</p> <ul style="list-style-type: none"> <li>▶ Group 1: Buttons               <ol style="list-style-type: none"> <li>0. OK only</li> <li>1. OK, Cancel</li> <li>2. Abort, Retry, Ignore</li> <li>3. Yes, No, Cancel</li> <li>4. Yes, No</li> <li>5. Retry, Cancel</li> </ol> </li> <li>▶ Group 2: Icons               <ol style="list-style-type: none"> <li>16. Critical Message ( STOP )</li> <li>32. Warning Query ( ? )</li> <li>48. Warning Message ( ! )</li> <li>64. Information Message ( i )</li> </ol> </li> <li>▶ Group 3: Defaults               <ol style="list-style-type: none"> <li>0. First button</li> <li>256. Second button</li> <li>512. Third button</li> </ol> </li> </ul> <p>If <i>buttons</i>% is omitted, MsgBox displays a single OK button.</p> |
| <i>title</i> \$  | A string expression containing the title for the message box.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |

Comments *Prompt*\$ does not accept strings of more than 1,023 characters.

After the user clicks a button, MsgBox returns a value indicating the user's choice. The return values for the MsgBox function are:

| Value | Button Pressed |
|-------|----------------|
| 1     | OK             |
| 2     | Cancel         |
| 3     | Abort          |
| 4     | Retry          |
| 5     | Ignore         |
| 6     | Yes            |
| 7     | No             |

## MsgBox (Function)

**Example**      This example displays one of each type of message box.

```
Sub main

 Dim btngrp as Integer
 Dim icongrp as Integer
 Dim defgrp as Integer
 Dim msgtext as String

 icongrp=16
 defgrp=0
 btngrp=0

 Do Until btngrp=6

 Select Case btngrp
 Case 1, 4, 5
 defgrp=0
 Case 2
 defgrp=256
 Case 3
 defgrp=512
 End Select

 msgtext=" Icon group = " & icongrp & Chr(10)
 msgtext=msgtext + " Button group = " & btngrp &
 Chr(10)
 msgtext=msgtext + " Default group = " & defgrp &
 Chr(10)
 msgtext=msgtext + Chr(10) + " Continue?"
 answer = MsgBox(msgtext, btngrp+icongrp+defgrp)

 Select Case answer
 Case 2,3,7
 Exit Do
 End Select

 If icongrp<>64 then
 icongrp=icongrp+16
 End If

 btngrp=btngrp+1

 Loop

End Sub
```

**See Also**      Dialog Boxes      MsgBox statement  
                  InputBox            PasswordBox



## MsgBox

### Statement

---

**Description** Displays a prompt in a message box.

**Syntax** **MsgBox** *prompt*\$, [*buttons*%][, *title*\$]

| Syntax Element   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>prompt</i> \$ | The text to display in a dialog box.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <i>buttons</i> % | <p>An integer value for the buttons, the icon, and the default button choice to display in a dialog box. <i>buttons</i>% is the sum of three values, one from each of the following groups:</p> <ul style="list-style-type: none"> <li>▶ <b>Group 1: Buttons</b> <ul style="list-style-type: none"> <li>0. OK only</li> <li>1. OK, Cancel</li> <li>2. Abort, Retry, Ignore</li> <li>3. Yes, No, Cancel</li> <li>4. Yes, No</li> <li>5. Retry, Cancel</li> </ul> </li> <li>▶ <b>Group 2: Icons</b> <ul style="list-style-type: none"> <li>16. Critical Message ( STOP )</li> <li>32. Warning Query ( ? )</li> <li>48. Warning Message ( ! )</li> <li>64. Information Message ( i )</li> </ul> </li> <li>▶ <b>Group 3: Defaults</b> <ul style="list-style-type: none"> <li>0. First button</li> <li>256. Second button</li> <li>512. Third button</li> </ul> </li> </ul> <p>If <i>buttons</i>% is omitted, MsgBox displays a single OK button.</p> |
| <i>title</i> \$  | A string expression containing the title for the message box.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

**Comments** *Prompt*\$ does not accept strings of more than 1,023 characters.

Name

Example

This example finds the future value of an annuity, whose terms are defined by the user. It uses the `MsgBox` statement to display the result.

```
Sub main
 Dim aprate, periods
 Dim payment, annuitypv
 Dim due, futurevalue
 Dim msgtext
 annuitypv=InputBox("Enter present value of the annuity: ")
 aprate=InputBox("Enter the annual percentage rate: ")
 If aprate > 1 then
 Aprate = aprate/100
 End If
 periods=InputBox("Enter the total number of pay periods: ")
 payment=InputBox("Enter the initial amount paid to you: ")
 Rem Assume payments are made at end of month
 due=0
 futurevalue=FV(aprate/12,periods,-payment,- annuitypv,due)
 msgtext="The future value is: " & Format(futurevalue,"Currency")
 MsgBox msgtext
End Sub
```

See Also

`InputBox`  
`MsgBox` function  
`PasswordBox`

## Name

### Statement

---

**Description** Renames a file or moves a file from one directory to another.

**Syntax** **Name** *oldfilename\$* As *newfilename\$*

| Syntax Element       | Description                                           |
|----------------------|-------------------------------------------------------|
| <i>oldfilename\$</i> | A string expression containing the file to rename.    |
| <i>newfilename\$</i> | A string expression containing the name for the file. |

**Comments** A path can be part of either file name argument. If the paths are different, the file is moved to the new directory.

A file must be closed in order to be renamed. If the file *oldfilename\$* is open or if the file *newfilename\$* already exists, SQABasic generates an error message.

**Example** This example creates a temporary file, C:\TEMP001, renames the file to C:\TEMP002, then deletes them both. It calls the sub procedure CREATEFILE to create the C:\TEMP001 file.

```

Declare Sub createfile()
Sub main
 Call createfile
 On Error Resume Next
 Name "C:\TEMP001" As "C:\TEMP002"
 MsgBox "The file has been renamed"
 MsgBox "Now deleting both files"
 Kill "TEMP001"
 Kill "TEMP002"
End Sub

Sub createfile()
 Rem Put the numbers 1-10 into a file
 Dim x as Integer
 Dim y()
 Dim startletter
 Open "C:\TEMP001" for Output as #1
 For x=1 to 10
 Write #1, x
 Next x
 Close #1
End Sub

```

**See Also** FileAttr      GetAttr  
FileCopy      Kill

## New Operator

---

**Description** Allocates and initializes a new OLE2 object of the named class.

**Syntax** Set *objectVar* = **New** *className*  
Dim *objectVar* As **New** *className*

| Syntax Element   | Description                                 |
|------------------|---------------------------------------------|
| <i>objectVar</i> | The OLE2 object to allocate and initialize. |
| <i>className</i> | The class to assign to the object.          |

**Comments** In the Dim statement, New marks *objectVar* so that a new object will be allocated and initialized when *objectVar* is first used. If *objectVar* is not referenced, then no new object will be allocated.

**Note:** An object variable that was declared with New will allocate a second object if *objectVar* is Set to Nothing and referenced again.

'\$NoCStrings

Example        None.

See Also       Dim        Set  
                 Global    Static

## '\$NoCStrings

Metacommand

»»SQA»

Description    Tells the compiler to treat a backslash (\) inside a string as a normal character.

Syntax        **'\$NoCStrings** [Save]

| Syntax Element | Description                                                                                                      |
|----------------|------------------------------------------------------------------------------------------------------------------|
| Save           | Saves the current '\$CStrings setting before restoring the treatment of the backslash (\) to a normal character. |

Comments      Use the '\$CStrings Restore command to restore a previously saved setting. Save and Restore operate as a stack and allow the user to change the '\$CStrings setting for a range of the program without impacting the rest of the program.

Use the '\$CStrings metacommand to tell the compiler to treat a backslash (\) inside of a string as an Escape character.

All metacommands must begin with an apostrophe ( ' ) and are recognized by the compiler only if the command starts at the beginning of a line.

Example        This example displays two lines, the first time using the C-language characters \n for a carriage return and line feed.

```
Sub main
 '$CStrings
 MsgBox "This is line 1\n This is line 2 (using C
 Strings)"
 '$NoCStrings
 MsgBox "This is line 1" +Chr$(13)+Chr$(10)+"This
 is line 2 (using Chr)"
End Sub
```

See Also       '\$CStrings  
                 '\$Include  
                 Rem

## Nothing

### Function

---

**Description** Returns an object value that does not refer to an object.

**Syntax** Set *variableName* = **Nothing**

| Syntax Element      | Description                                        |
|---------------------|----------------------------------------------------|
| <i>variableName</i> | The name of the object variable to set to nothing. |

**Comments** Nothing is the value object variables have when they do not refer to an object, either because they have not been initialized yet or because they were explicitly Set to Nothing. For example:

```
If Not objectVar Is Nothing then
 objectVar.Close
 Set objectVar = Nothing
End If
```

**Example** This example displays a list of open files in the software application VISIO. It uses the Nothing function to determine whether VISIO is available. To see how this example works, you need to start VISIO and open one or more documents.

```
Sub main
 Dim visio as Object
 Dim doc as Object
 Dim msgtext as String
 Dim i as Integer, doccount as Integer

 'Initialize Visio
 ' find Visio
 Set visio = GetObject(,"visio.application")
 If (visio Is Nothing) then
 MsgBox "Couldn't find Visio!"
 Exit Sub
 End If
 'Get # of open Visio files
 'OLE2 call to Visio
 doccount = visio.documents.count
 If doccount=0 then
 msgtext="No open Visio documents."
 Else
 msgtext="The open files are: " & Chr$(13)
 For i = 1 to doccount
 ' access Visio's document method
 Set doc = visio.documents(i)
 msgtext=msgtext & Chr$(13) & doc.name
 Next i
 End If
 MsgBox msgtext
End Sub
```

**See Also** Is  
New

Now

## Now Function

---

**Description** Returns the current date and time.

**Syntax** **Now( )**

**Comments** The **Now** function returns a Variant of VarType 7 (date) that represents the current date and time according to the setting of the computer's system date and time.

**Example** This example finds the month (1-12) and day (1-31) values for this Thursday.

```
Sub main
 Dim x, today
 Dim msgtext
 Today=DateValue(Now)
 Let x=0
 Do While Weekday(Today+x) <> 5
 x=x+1
 Loop
 msgtext="This Thursday is: " & Month(Today+x)&"/"&Day(Today+x)
 MsgBox msgtext
End Sub
```

**See Also**

|                |               |                |
|----------------|---------------|----------------|
| Date function  | Minute        | Time statement |
| Date statement | Month         | Weekday        |
| Day            | Second        | Year           |
| Hour           | Time function |                |

## NPV Function

---

**Description** Returns the net present value of an investment based on a stream of periodic cash flows and a constant interest rate.

**Syntax** **NPV** (*rate*, *valuearray*())

| Syntax Element       | Description                           |
|----------------------|---------------------------------------|
| <i>rate</i>          | Discount rate per period.             |
| <i>valuearray</i> () | An array containing cash flow values. |

**Comments** `Valuearray()` must have at least one positive value (representing a receipt) and one negative value (representing a payment). All payments and receipts must be represented in the exact sequence. The value returned by `NPV` will vary with the change in the sequence of cash flows.

If the discount rate is 12% per period, `rate` is the decimal equivalent, i.e. 0.12.

`NPV` uses future cash flows as the basis for the net present value calculation. If the first cash flow occurs at the beginning of the first period, its value should be added to the result returned by `NPV` and must not be included in `valuearray()`.

**Example** This example finds the net present value of an investment, given a range of cash flows by the user.

```
Sub main
 Dim aprate as Single
 Dim varray() as Double
 Dim cflowper as Integer
 Dim x as Integer
 Dim netpv as Double
 cflowper=InputBox("Enter number of cash flow periods")
 ReDim varray(cflowper)
 For x= 1 to cflowper
 varray(x)=InputBox("Cash flow amount for period #" & x & ":")
 Next x
 aprate=InputBox("Enter discount rate: ")
 If aprate>1 then
 aprate=aprate/100
 End If
 netpv=NPV(aprate,varray())
 MsgBox "The net present value is: " & Format(netpv, "Currency")
End Sub
```

**See Also**

|      |      |
|------|------|
| FV   | PPmt |
| IPmt | PV   |
| IRR  | Rate |
| Pmt  |      |

## Null

### Function

---

**Description** Returns a Variant value set to NULL.

**Syntax** `Null`

## Object Class

Comments `Null` is used to set a Variant to the `Null` value explicitly, as follows:

```
variableName = Null
```

Note that Variants are initialized by SQABasic to the `empty` value, which is different from the `null` value.

Example This example asks for ten test score values and calculates the average. If any score is negative, the value is set to `Null`. Then `IsNull` is used to reduce the total count of scores (originally 10) to just those with positive values before calculating the average.

```
Sub main
 Dim arrayvar(10)
 Dim count as Integer
 Dim total as Integer
 Dim x as Integer
 Dim msgtext as String
 Dim tscore as Single
 count=10
 total=0
 For x=1 to count
 tscore=InputBox("Enter test score #" & x & ":")
 If tscore<0 then
 arrayvar(x)=Null
 Else
 arrayvar(x)=tscore
 total=total+arrayvar(x)
 End If
 Next x
 Do While x<>0
 x=x-1
 If IsNull(arrayvar(x))=-1 then
 count=count-1
 End If
 Loop
 msgtext="Average (excluding negative values) is: " & Chr(10)
 msgtext=msgtext & Format (total/count, "##.##")
 MsgBox msgtext
End Sub
```

See Also `IsEmpty`  
`IsNull`  
`VarType`

## Object Class

---

Description A class that provides access to OLE2 automation objects.



Syntax `Dim variableName As Object`

| Syntax Element      | Description                                 |
|---------------------|---------------------------------------------|
| <i>variableName</i> | The name of the object variable to declare. |

Comments To create a new object, first dimension a variable, using the `Dim` statement, then `Set` the variable to the return value of `CreateObject` or `GetObject`, as follows:

```
Dim OLE2 As Object
SetOLE2 = CreateObject("spoly.cpoly")
```

To refer to a method or property of the newly created object, use the syntax: *objectvar.property* or *objectvar.method*, as follows:

```
OLE2.reset
```

Example This example displays a list of open files in the software application VISIO. It uses the `Object` class to declare the variables used for accessing VISIO and its document files and methods.

```
Sub main
 Dim visio as Object
 Dim doc as Object
 Dim msgtext as String
 Dim i as Integer, doccount as Integer

 'Initialize Visio
 ' find Visio
 Set visio = GetObject(,"visio.application")
 If (visio Is Nothing) then
 MsgBox "Couldn't find Visio!"
 Exit Sub
 End If
 'Get # of open Visio files
 'OLE2 call to Visio
 doccount = visio.documents.count
 If doccount=0 then
 msgtext="No open Visio documents."
 Else
 msgtext="The open files are: " & Chr$(13)
 For i = 1 to doccount
 ' access Visio's document method
 Set doc = visio.documents(i)
 msgtext=msgtext & Chr$(13) & doc.name
 Next i
 End If
 MsgBox msgtext
End Sub
```

See Also

|               |         |
|---------------|---------|
| Class List    | New     |
| Create Object | Nothing |
| Get Object    | Typeof  |

Oct

## Oct Function

---

**Description** Returns the octal representation of a number, as a string.

**Syntax** `Oct [ $ ] ( number )`

| Syntax Element | Description                                                                                                                                            |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| \$             | Optional. If specified the return data type is <code>String</code> . If omitted the function will return a variant of <code>VarType 8</code> (string). |
| <i>number</i>  | A numeric expression for the number to convert to octal.                                                                                               |

**Comments** If the numeric expression has a data type of `Integer`, the string contains up to six octal digits; otherwise, the expression will be converted to a data type of `Long`, and the string can contain up to 11 octal digits.

To represent an octal number directly, precede the octal value with `&O`. For example, `&O10` equals decimal 8 in octal notation.

**Example** This example prints the octal values for the numbers from 1 to 15.

```
Sub main
 Dim x,y
 Dim msgtext
 Dim nofspace
 msgtext="Octal numbers from 1 to 15:" & Chr(10)
 For x=1 to 15
 nofspace=10
 y=Oct(x)
 If Len(x)=2 then
 nofspace=nofspace-2
 End If
 msgtext=msgtext & Chr(10) & x & Space(nospace) & y
 Next x
 MsgBox msgtext
End Sub
```

**See Also** [Hex](#)

## OKButton

### Statement

---

**Description** Determines the position and size of an OK button in a dialog box.

**Syntax** `OKButton x, y, dx, dy[, .id]`

| Syntax Element      | Description                                                                        |
|---------------------|------------------------------------------------------------------------------------|
| <code>x, y</code>   | The position of the OK button relative to the upper left corner of the dialog box. |
| <code>dx, dy</code> | The width and height of the button.                                                |
| <code>.id</code>    | An optional identifier for the button.                                             |

**Comments** A `dy` value of 14 typically accommodates text in the system font.  
`.id` is an optional identifier used by the dialog statements that act on this control.  
 Use the `OKButton` statement only between a `Begin Dialog` and an `End Dialog` statement.

**Example** This example defines a dialog box with a `DropComboBox` and the OK and Cancel buttons.

```
Sub main
 Dim cchoices as String
 On Error Resume Next
 cchoices="All"+Chr$(9)+"Nothing"
 Begin Dialog UserDialog 180, 95, "SQABasic Dialog Box"
 ButtonGroup .ButtonGroup1
 Text 9, 3, 69, 13, "Filename:", .Text1
 DropComboBox 9, 17, 111, 41, cchoices, .ComboBox1
 OKButton 131, 8, 42, 13
 CancelButton 131, 27, 42, 13
 End Dialog
 Dim mydialogbox As UserDialog
 Dialog mydialogbox
 If Err=102 then
 MsgBox "You pressed Cancel."
 Else
 MsgBox "You pressed OK."
 End If
End Sub
```

**See Also**

|                               |                           |                             |
|-------------------------------|---------------------------|-----------------------------|
| <code>Begin/End Dialog</code> | <code>ComboBox</code>     | <code>OptionGroup</code>    |
| <code>Button</code>           | <code>Dialog</code>       | <code>Picture</code>        |
| <code>ButtonGroup</code>      | <code>DropComboBox</code> | <code>StaticComboBox</code> |
| <code>CancelButton</code>     | <code>GroupBox</code>     | <code>Text</code>           |
| <code>Caption</code>          | <code>ListBox</code>      | <code>TextBox</code>        |
| <code>CheckBox</code>         | <code>OptionButton</code> |                             |

On...GoTo

## On...GoTo

Statement

---

**Description** Branch to a label in the current procedure based on the value of a numeric expression.

**Syntax** `ON numeric-expression GoTo label1[, label2,... ]`

| Syntax Element                  | Description                                                                                                    |
|---------------------------------|----------------------------------------------------------------------------------------------------------------|
| <code>numeric-expression</code> | Any numeric expression that evaluates to a positive number.                                                    |
| <code>label1, label2</code>     | A label in the current procedure to branch to if <code>numeric-expression</code> evaluates to 1, 2, and so on. |

**Comments** If `numeric expression` evaluates to 0 or to a number greater than the number of labels following `GoTo`, the program continues at the next statement. If `numeric-expression` evaluates to a number less than 0 or greater than 255, an Illegal function call error is issued.

A label has the same format as any other SQABasic name. See Appendix A for more information about SQABasic labels and names.

**Example** This example sets the current system time to the user's entry. If the entry cannot be converted to a valid time value, this sub procedure sets the variable to Null. It then checks the variable and if it is Null, uses the `On . . . GoTo` statement to ask again.

```
Sub main
 Dim answer as Integer
 answer=InputBox("Enter a choice (1-3) or 0 to quit")
 On answer GoTo c1, c2, c3
 MsgBox("You typed 0.")
 Exit Sub
c1: MsgBox("You picked choice 1.")
 Exit Sub
c2: MsgBox("You picked choice 2.")
 Exit Sub
c3: MsgBox("You picked choice 3.")
 Exit Sub
End Sub
```

**See Also** Goto  
Select Case

## On Error Statement

---

**Description** Specifies the location of an error-handling routine within the current procedure.

**Syntax** **ON** [*Local*] **Error** {*GoTo label* [*Resume Next*]  
*GoTo 0*}

| Syntax Element | Description                                                                                          |
|----------------|------------------------------------------------------------------------------------------------------|
| <i>label</i>   | A string used as a label in the current procedure to identify the lines of code that process errors. |

**Comments** *On Error* can also be used to disable an error-handling routine. Unless an *On Error* statement is used, any runtime error will be fatal (SQABasic will terminate the execution of the program).

An *On Error* statement is composed of the following parts:

| Part               | Definition                                                                                                                                                                                                                     |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Local</i>       | Keyword allowed in error-handling routines at the procedure level. Used to ensure compatibility with other Variants of SQABasic.                                                                                               |
| <i>GoTo label</i>  | Enables the error-handling routine that starts at <i>label</i> . If the designated label is not in the same procedure as the <i>On Error</i> statement, SQABasic generates an error message.                                   |
| <i>Resume Next</i> | Designates that error-handling code is handled by the statement that immediately follows the statement that caused an error. At this point, use the <code>Err</code> function to retrieve the error-code of the runtime error. |
| <i>GoTo 0</i>      | Disables any error handler that has been enabled.                                                                                                                                                                              |

When it is referenced by an *On Error GoTo label* statement, an error-handler is enabled. Once this enabling occurs, a runtime error will result in program control switching to the error-handling routine and “activating” the error handler. The error handler remains active from the time the runtime error has been trapped until a *Resume* statement is executed in the error handler.

If another error occurs while the error handler is active, SQABasic will search for an error handler in the procedure that called the current procedure (if this fails, SQABasic will look for a handler belonging to the caller’s caller, and so on). If a handler is found, the current procedure will terminate, and the error handler in the calling procedure will be activated.

## Open

It is an error (No Resume) to execute an End Sub or End Function statement while an error handler is active. The Exit Sub or Exit Function statement can be used to end the error condition and exit the current procedure.

A label has the same format as any other SQABasic name. See Appendix A for more information about SQABasic labels and names.

### Example

This example prompts the user for a drive and directory name and uses On Error to trap invalid entries.

```
Sub main
 Dim userdrive, userdir, msgtext
in1: userdrive=InputBox("Enter drive:",,"C:")
 On Error Resume Next
 ChDrive userdrive
 If Err=68 then
 MsgBox "Invalid Drive. Try again."
 Goto in1
 End If
in2: On Error Goto Errhdlr1
 userdir=InputBox("Enter directory path:")
 ChDir userdrive & userdir
 MsgBox "New default directory is: " & userdrive & userdir
 Exit Sub
Errhdlr1:
 Select Case Err
 Case 75
 msgtext="Path is invalid."
 Case 76
 msgtext="Path not found."
 Case 70
 msgtext="Permission denied."
 Case Else
 msgtext="Error " & Err & ": " & Error$ & "occurred."
 End Select
 MsgBox msgtext & " Try again."
 Resume in2
End Sub
```

### See Also

|               |                 |
|---------------|-----------------|
| Erl           | Error function  |
| Err function  | Error statement |
| Err statement | Resume          |

## Open Statement

---

Description    Opens a file or device for input or output.

## Syntax

**Open** *filename*\$ [For *mode*] [Access *access*] [*lock*] As  
[*#*]*filename*% [Len = *reclen*]

| Syntax Element     | Description                                                                                                                                                                                                                                                                                                                                                 |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>filename</i> \$ | A string or string expression for the name of the file to open.                                                                                                                                                                                                                                                                                             |
| <i>mode</i>        | One of the following keywords: <ul style="list-style-type: none"> <li>▶ Input. Read data from the file sequentially.</li> <li>▶ Output. Put data into the file sequentially.</li> <li>▶ Append. Add data to the file sequentially.</li> <li>▶ Random. Get data from the file by random access.</li> <li>▶ Binary. Get binary data from the file.</li> </ul> |
| <i>access</i>      | One of the following keywords: <ul style="list-style-type: none"> <li>▶ Read. Read data from the file only.</li> <li>▶ Write. Write data to the file only.</li> <li>▶ Read Write. Read or write data to the file.</li> </ul>                                                                                                                                |
| <i>lock</i>        | One of the following keywords to designate access by other processes: <ul style="list-style-type: none"> <li>▶ Shared. Read or write available on the file.</li> <li>▶ Lock Read. Read data only.</li> <li>▶ Lock Write. Write data only.</li> <li>▶ Lock Read Write. No read or write available.</li> </ul>                                                |
| <i>filename</i> %  | An integer or expression containing the integer to assign to the open file (between 1 and 255).                                                                                                                                                                                                                                                             |
| <i>reclen</i>      | The length of the records (for Random or Binary files only).                                                                                                                                                                                                                                                                                                |

## Comments

A file must be opened before any input/output operation can be performed on it.

If *filename*\$ does not exist, it is created when opened in Append, Binary, Output or Random modes.

If *mode* is not specified, it defaults to Random.

If *access* is not specified for Random or Binary modes, *access* is attempted in the following order: Read Write, Write, Read.

If *lock* is not specified, *filename*\$ can be opened by other processes that do not specify a *lock*, although that process cannot perform any file operations on the file while the original process still has the file open.

Use the FreeFile function to find the next available value for *filename*%.

*Reclen* is ignored for Input, Output, and Append modes.

## Option Base

**Example** This example opens a file for Random access, gets the contents of the file, and closes the file again. The second sub procedure, CREATEFILE, creates the file C:\TEMP001 used by the main sub procedure.

```
Declare Sub createfile()
Sub main
 Dim acctno as String*3
 Dim recno as Long
 Dim msgtext as String
 Dim newline as String
 Call createfile
 recno=1
 newline=Chr(10)
 Open "C:\TEMP001" For Random As #1 Len=3
 msgtext="The account numbers are:" & newline
 Do Until recno=11
 Get #1,recno,acctno
 msgtext=msgtext & acctno
 recno=recno+1
 Loop
 MsgBox msgtext
 Close #1
 Kill "C:\TEMP001"
End Sub

Sub createfile()
 Rem Put the numbers 1-10 into a file
 Dim x as Integer
 Open "C:\TEMP001" for Output as #1
 For x=1 to 10
 Write #1, x
 Next x
 Close #1
End Sub
```

**See Also** Close  
FreeFile

## Option Base Statement

---

**Description** Specifies the default lower bound to use for array subscripts.

**Syntax** **Option Base** *lowerBound%*

| Syntax Element     | Description                                                                            |
|--------------------|----------------------------------------------------------------------------------------|
| <i>lowerBound%</i> | A number or expression containing a number for the default lower bound: either 0 or 1. |



- Comments** If no `Option Base` statement is specified, the default lower bound for array subscripts will be 0.
- The `Option Base` statement is *not* allowed inside a procedure, and must precede any use of arrays in the module. Only one `Option Base` statement is allowed per module.
- Example** This example resizes an array if the user enters more data than can fit in the array. It uses `LBound` and `UBound` to determine the existing size of the array and `ReDim` to resize it. `Option Base` sets the default lower bound of the array to 1.

```

Option Base 1
Sub main

 Dim arrayvar() as Integer
 Dim count as Integer
 Dim answer as String
 Dim x, y as Integer
 Dim total

 total=0
 x=1
 count=InputBox("How many test scores do you have?")
 ReDim arrayvar(count)

start:
 Do until x=count+1
 arrayvar(x)=InputBox("Enter test score #" &x & ":")
 x=x+1
 Loop

 answer=InputBox$("Do you have more scores? (Y/N)")

 If answer="Y" or answer="y" then
 count=InputBox("How many more do you have?")
 If count<>0 then
 count=count+(x-1)
 ReDim Preserve arrayvar(count)
 Goto start
 End If
 End If

 x=LBound(arrayvar,1)
 count=UBound(arrayvar,1)
 For y=x to count
 total=total+arrayvar(y)
 Next y

 MsgBox "The average of " & count & " scores is " & Int(total/count)

End Sub

```

- See Also**
- |                     |                     |
|---------------------|---------------------|
| <code>Dim</code>    | <code>ReDim</code>  |
| <code>Global</code> | <code>Static</code> |
| <code>LBound</code> |                     |

## Option Compare

### Statement

---

**Description** Specifies the default method for string comparisons: either case-sensitive or case-insensitive.

**Syntax** `Option Compare { Binary | Text }`

| Syntax Element | Description                                                                     |
|----------------|---------------------------------------------------------------------------------|
| Binary         | Comparisons are case-sensitive (lowercase and uppercase letters are different). |
| Text           | Comparisons are not case-sensitive.                                             |

**Comments** Binary comparisons compare strings based upon the ANSI character set. Text comparisons are based upon the relative order of characters as determined by the country code setting for your system.

**Example** This example compares two strings: "Jane Smith" and "jane smith". When `Option Compare` is `Text`, the strings are considered the same. If `Option Compare` is `Binary`, they will not be the same. `Binary` is the default. To see the difference, run the example once, and then run it again, commenting out the `Option Compare` statement.

```

Option Compare Text
Sub main
 Dim strg1 as String
 Dim strg2 as String
 Dim retvalue as Integer
 strg1="JANE SMITH"
 strg2="jane smith"
 i:
 retvalue=StrComp(strg1,strg2)
 If retvalue=0 then
 MsgBox "The strings are identical"
 Else
 MsgBox "The strings are not identical"
 End If
 Exit Sub
 End Sub

```

**See Also** `Instr`  
`StrComp`

## Option Explicit

### Statement

---

|             |                                                                                                                                                                                                                                                                                              |         |              |     |         |        |         |     |       |  |          |        |  |
|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|--------------|-----|---------|--------|---------|-----|-------|--|----------|--------|--|
| Description | Specifies that all variables in a module <i>must</i> be explicitly declared.                                                                                                                                                                                                                 |         |              |     |         |        |         |     |       |  |          |        |  |
| Syntax      | <b>Option Explicit</b>                                                                                                                                                                                                                                                                       |         |              |     |         |        |         |     |       |  |          |        |  |
| Comments    | By default, SQABasic automatically declares any variables that do not appear in a Dim, Global, Redim, or Static statement. Option Explicit causes such variables to produce a Variable Not Declared error.                                                                                   |         |              |     |         |        |         |     |       |  |          |        |  |
| Example     | <p>This example specifies that all variables must be explicitly declared, thus preventing any mistyped variable names.</p> <pre> Option Explicit Sub main     Dim counter As Integer     Dim fixedstring As String*25     Dim varstring As String     ...           Code here End Sub </pre> |         |              |     |         |        |         |     |       |  |          |        |  |
| See Also    | <table> <tr> <td>Const</td> <td>End function</td> <td>Sub</td> </tr> <tr> <td>Deftype</td> <td>Global</td> <td>End Sub</td> </tr> <tr> <td>Dim</td> <td>ReDim</td> <td></td> </tr> <tr> <td>Function</td> <td>Static</td> <td></td> </tr> </table>                                           | Const   | End function | Sub | Deftype | Global | End Sub | Dim | ReDim |  | Function | Static |  |
| Const       | End function                                                                                                                                                                                                                                                                                 | Sub     |              |     |         |        |         |     |       |  |          |        |  |
| Deftype     | Global                                                                                                                                                                                                                                                                                       | End Sub |              |     |         |        |         |     |       |  |          |        |  |
| Dim         | ReDim                                                                                                                                                                                                                                                                                        |         |              |     |         |        |         |     |       |  |          |        |  |
| Function    | Static                                                                                                                                                                                                                                                                                       |         |              |     |         |        |         |     |       |  |          |        |  |

## OptionButton

### Statement

---

|             |                                                                                 |
|-------------|---------------------------------------------------------------------------------|
| Description | Defines the position and text associated with an option button in a dialog box. |
| Syntax      | <b>OptionButton</b> <i>x, y, dx, dy, text\$[, .id]</i>                          |

| Syntax Element | Description                                                                                                                               |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| <i>x, y</i>    | The position of the button relative to the upper left corner of the dialog box.                                                           |
| <i>dx, dy</i>  | The width and height of the button.                                                                                                       |
| <i>text\$</i>  | A string to display next to the option button. If the width of this string is greater than <i>dx</i> , trailing characters are truncated. |
| <i>.id</i>     | An optional identifier used by the dialog statements that act on this control.                                                            |

## OptionGroup

**Comments** You must have at least two `OptionButton` statements in a dialog box. You use these statements in conjunction with the `OptionGroup` statement.

A *dy* value of 12 typically accommodates text in the system font.

To enable the user to select an option button by typing a character from the keyboard, precede the character in *text\$* with an ampersand (&).

Use the `OptionButton` statement only between a `Begin Dialog` and an `End Dialog` statement.

**Example** This example creates a dialog box with a group box with two option buttons: All pages and Range of pages.

```
Sub main
 Begin Dialog UserDialog 183, 70, "SQABasic Dialog Box"
 GroupBox 5, 4, 97, 57, "File Range"
 OptionGroup .OptionGroup2
 OptionButton 16, 12, 46, 12, "All pages", .OptionButton3
 OptionButton 16, 28, 67, 8, "Range of pages", .OptionButton4
 Text 22, 39, 20, 10, "From:", .Text6
 Text 60, 39, 14, 9, "To:", .Text7
 TextBox 76, 39, 13, 12, .TextBox4
 TextBox 44, 39, 12, 11, .TextBox5
 OKButton 125, 6, 54, 14
 CancelButton 125, 26, 54, 14
 End Dialog
 Dim mydialog as UserDialog
 On Error Resume Next
 Dialog mydialog
 If Err=102 then
 MsgBox "Dialog box canceled."
 End If
End Sub
```

|                 |                               |                           |                             |
|-----------------|-------------------------------|---------------------------|-----------------------------|
| <b>See Also</b> | <code>Begin/End Dialog</code> | <code>ComboBox</code>     | <code>OptionGroup</code>    |
|                 | <code>Button</code>           | <code>Dialog</code>       | <code>Picture</code>        |
|                 | <code>ButtonGroup</code>      | <code>DropComboBox</code> | <code>StaticComboBox</code> |
|                 | <code>CancelButton</code>     | <code>GroupBox</code>     | <code>Text</code>           |
|                 | <code>Caption</code>          | <code>Listbox</code>      | <code>TextBox</code>        |
|                 | <code>CheckBox</code>         | <code>OKButton</code>     |                             |

## OptionGroup Statement

---

**Description** Groups a series of option buttons under one heading in a dialog box.

Syntax **OptionGroup** *.field*

| Syntax Element | Description                                                                                                            |
|----------------|------------------------------------------------------------------------------------------------------------------------|
| <i>.field</i>  | A value for the option button selected by the user: 0 for the first option button, 1 for the second button, and so on. |

Comments The `OptionGroup` statement is used in conjunction with `OptionButton` statements to set up a series of related options. The `OptionGroup` statement begins the definition of the option buttons and establishes the dialog-record field that will contain the option selection.

Use the `OptionGroup` statement only between a `Begin Dialog` and an `End Dialog` statement.

Example This example creates a dialog box with a group box with two option buttons: All Pages and Range of Pages.

```
Sub main
 Begin Dialog UserDialog 192, 71, "SQABasic Dialog Box"
 GroupBox 7, 6, 97, 57, "File Range"
 OptionGroup .OptionGroup2
 OptionButton 18, 14, 46, 12, "All Pages", .OptionButton3
 OptionButton 18, 30, 67, 8, "Range of Pages", .OptionButton4
 Text 24, 41, 20, 10, "From:", .Text6
 Text 62, 41, 14, 9, "To:", .Text7
 TextBox 78, 41, 13, 12, .TextBox4
 TextBox 46, 41, 12, 11, .TextBox5
 OKButton 126, 6, 54, 14
 CancelButton 126, 26, 54, 14
 End Dialog

 Dim mydialog as UserDialog
 On Error Resume Next
 Dialog mydialog

 If Err=102 then
 MsgBox "Dialog box canceled."
 End If

End Sub
```

|          |                               |                           |                             |
|----------|-------------------------------|---------------------------|-----------------------------|
| See Also | <code>Begin/End Dialog</code> | <code>ComboBox</code>     | <code>OptionButton</code>   |
|          | <code>Button</code>           | <code>Dialog</code>       | <code>Picture</code>        |
|          | <code>ButtonGroup</code>      | <code>DropComboBox</code> | <code>StaticComboBox</code> |
|          | <code>CancelButton</code>     | <code>GroupBox</code>     | <code>Text</code>           |
|          | <code>Caption</code>          | <code>ListBox</code>      | <code>TextBox</code>        |
|          | <code>CheckBox</code>         | <code>OKButton</code>     |                             |

# Pager

## User Action Command

**Description** Performs an action on a Pager control.

**Syntax** **Pager** *action%*, *recMethod\$*, *parameters\$*

| Syntax Element      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i>      | <p>One of these mouse actions:</p> <ul style="list-style-type: none"> <li>▶ <i>MouseClicked</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> must contain <i>Coords=x,y</i>.</li> <li>▶ <i>MouseDown</i>. The dragging of the mouse while mouse buttons and/or shifting keys (Ctrl, Alt, Shift) are pressed. When <i>action%</i> contains a mouse-drag value, <i>parameters\$</i> must contain <i>Coords=x1,y1,x2,y2</i>.</li> </ul> <p>See Appendix E for a list of mouse click and drag values.</p>                                                                                                                                                      |
| <i>recMethod\$</i>  | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <i>ID=%</i>. The object's internal Windows ID.</li> <li>▶ <i>Label=\$</i>. The text of the label object that immediately precedes the control in the Z order of windows.</li> <li>▶ <i>Name=\$</i>. A unique name that a developer assigns to an object to identify the object in the development environment. For example, the object name for a command button might be <i>Command1</i>.</li> <li>▶ <i>ObjectIndex=%</i>. The number of the object among all objects of the same type in the same window.</li> <li>▶ <i>Text=\$</i>. The text displayed on the object.</li> <li>▶ <i>VisualText=\$</i>. An optional setting used to identify an object by its prior label. It is for user clarification only and does not affect object recognition.</li> </ul> |
| <i>parameters\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <i>Coords=x,y</i>. If <i>action%</i> is a mouse click, specifies the coordinates of the click, relative to the top left of the object.</li> <li>▶ <i>Coords=x1,y1,x2,y2</i>. If <i>action%</i> is a mouse drag, specifies the coordinates, where <i>x1,y1</i> are the starting coordinates of the drag, and <i>x2,y2</i> are the ending coordinates. The coordinates are relative to the top left of the object.</li> </ul>                                                                                                                                                                                                                                                                                                                                       |

- Comments    None.
- Example      This example clicks the first pager control in the window (ObjectIndex=1) at x,y coordinates of 202,12.  
                  **Pager** Click, "ObjectIndex=1", "Coords=202,12"
- See Also      PagerVP

## PagerVP

Verification Point Command



Description    Establishes a verification point for a pager control.

Syntax          *Result* = **PagerVP** (*action%*, *recMethod\$*, *parameters\$*)

| Syntax Element     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i>     | The type of verification to perform. Valid values: <ul style="list-style-type: none"> <li>▶ CompareProperties. Captures object properties information for the object and compares it to a recorded baseline. <i>parameters\$ VP</i> is required; ExpectedResult and Wait are optional.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <i>recMethod\$</i> | Valid values: <ul style="list-style-type: none"> <li>▶ ID=%. The object's internal Windows ID.</li> <li>▶ Label=\$. The text of the label object that immediately precedes the control in the internal order (Z order) of windows.</li> <li>▶ Name=\$. A unique name that a developer assigns to an object to identify the object in the development environment. For example, the object name for a command button might be Command1.</li> <li>▶ ObjectIndex=%. The number of the object among all objects of the same type in the same window.</li> <li>▶ Text=\$. The text displayed on the object.</li> <li>▶ VisualText=\$. An optional setting used to identify an object by its prior label. It is for user clarification only and does not affect object recognition.</li> </ul> |



## PasswordBox

► ► ►

| Syntax Element       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>parameters</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> <li>► <b>ExpectedResult=%</b>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> <li>- <b>PASS</b>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail.</li> <li>- <b>FAIL</b>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail.</li> </ul> </li> <li>► <b>VP=\$</b>. The verification point ID. IDs must be unique within a script. Required for all verification points.</li> <li>► <b>Wait=%,%</b>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10,40</code> (retry the test every 10 seconds, but time out the test after 40 seconds).</li> </ul> |

**Comments** This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

**Example** This example captures the properties of the first pager control in the window (`ObjectIndex=1`) and compares them to the recorded baseline in verification point `PAGER1`.

```
Result = PagerVP (CompareProperties, "ObjectIndex=1", "VP=PAGER1")
```

**See Also** `Pager`

## PasswordBox

### Function

**Description** Returns a string entered by the user without echoing it to the screen.

**Syntax** `PasswordBox[$](prompt$, [title$], [default$][, xpos%, ypos%])`

| Syntax Element | Description                                                                                                                                                      |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| \$             | Optional. If specified the return type is <code>String</code> . If omitted, the function will return a <code>Variant</code> of <code>VarType 8 (String)</code> . |

► ► ►



▶ ▶ ▶

| Syntax Element              | Description                                                                      |
|-----------------------------|----------------------------------------------------------------------------------|
| <i>prompt\$</i>             | A string expression containing the text to show in the dialog box.               |
| <i>title\$</i>              | The caption for the dialog box's title bar.                                      |
| <i>default\$</i>            | The string expression shown in the edit box as the default response.             |
| <i>xpos%</i> , <i>ypos%</i> | The position of the dialog box, relative to the upper left corner of the screen. |

**Comments** The `PasswordBox` function displays a dialog box containing a prompt. Once the user has entered text, or made the button choice being prompted for, the contents of the box are returned.

The length of *prompt\$* is restricted to 255 characters. This figure is approximate and depends on the width of the characters used. Note that a carriage return and a line-feed character must be included in *prompt\$* if a multiple-line prompt is used.

If either *prompt\$* or *default\$* is omitted, nothing is displayed.

*Xpos%* determines the horizontal distance between the left edge of the screen and the left border of the dialog box, measured in dialog box units. *Ypos%* determines the horizontal distance from the top of the screen to the dialog box's upper edge, also in dialog box units. If these arguments are not entered, the dialog box is centered roughly one third of the way down the screen. A horizontal dialog box unit is 1/4 of the average character width in the system font; a vertical dialog box unit is 1/8 of the height of a character in the system font.

**Note:** To specify the dialog box's position, you must enter both of these arguments. If you enter one without the other, the default positioning is used.

Once the user presses Enter, or selects the OK button, `PasswordBox` returns the text contained in the password box. If the user selects Cancel, the `PasswordBox` function returns a null string ("").

**Example** This example asks the user for a password.

```
Sub main
 Dim retvalue
 Dim a
 retvalue=PasswordBox("Enter your login password",Password)
 If retvalue<>" " then
 MsgBox "Verifying password"
 ... 'Continue code here
 Else
 MsgBox "Login canceled"
 End If
End Sub
```

Picture

See Also       
                  

## Picture Statement

---

**Description**    Defines a picture control in a dialog box.

**Syntax**        **Picture** *x, y, dx, dy, filename\$, type[, .id]*

| Syntax Element    | Description                                                                             |
|-------------------|-----------------------------------------------------------------------------------------|
| <i>x, y</i>       | The position of the picture relative to the upper left corner of the dialog box.        |
| <i>dx, dy</i>     | The width and height of the picture.                                                    |
| <i>filename\$</i> | The name of the bitmap file (a file with .BMP extension) where the picture is located.  |
| <i>type</i>       | An integer for the location of the bitmap (0= <i>filename\$</i> , 3=Windows Clipboard). |
| <i>.id</i>        | An optional identifier used by the dialog statements that act on this control.          |

**Comments**     The `Picture` statement can only be used between a `Begin Dialog` and an `End Dialog` statement.

**Note:** The picture will be scaled equally in both directions and centered if the dimensions of the picture are not proportional to *dx* and *dy*.

If *type%* is 3, *filename\$* is ignored.

If the picture is not available (the file *filename\$* does not exist, does not contain a bitmap, or there is no bitmap on the Clipboard), the picture control will display the picture frame and the text (missing picture). This behavior can be changed by adding 16 to the value of *type%*. If *type%* is 16 or 19 and the picture is not available, a runtime error occurs.

**Example**        This example defines a dialog box with a picture along with the OK and Cancel buttons. The example assumes that your Windows directory is named Windows.

```
Sub main
 Begin Dialog UserDialog 148, 73, "SQABasic Dialog Box"
 Picture 8, 7, 46, 46, "C:\WINDOWS\CIRCLES.BMP", 0
 OKButton 80, 10, 54, 14
 CancelButton 80, 30, 54, 14
 End Dialog
```

```

Dim mydialog as UserDialog
On Error Resume Next
Dialog mydialog
If Err=102 then
 MsgBox "Dialog box canceled."
End If
End Sub

```

|          |                  |              |                |
|----------|------------------|--------------|----------------|
| See Also | Begin/End Dialog | ComboBox     | OptionButton   |
|          | Button           | Dialog       | OptionGroup    |
|          | ButtonGroup      | DropComboBox | StaticComboBox |
|          | CancelButton     | GroupBox     | Text           |
|          | Caption          | ListBox      | TextBox        |
|          | CheckBox         | OKButton     |                |

## PlayJrnl

Utility Command



**Description** Starts playback of a series of low-level recorded mouse and keyboard actions.

**Syntax** **PlayJrnl** *scriptID*

| Syntax Element  | Description                                                      |
|-----------------|------------------------------------------------------------------|
| <i>scriptID</i> | A unique number that Robot assigns to the low-level script file. |

**Comments** When you click Record → Turn Low-Level Recording On in Robot during recording, subsequent mouse and keyboard actions are automatically stored in an external file. Robot inserts the `PlayJrnl` command into the script to reference the external low-level file.

Low-level scripts are listed in the Robot Asset pane (to the left of the SQABasic script area of the Robot window). To display the contents of a low-level file, double-click the file's ID.

To return to Object-Oriented Recording, click Record → Turn Low-Level Recording Off.

**Example** This example plays back the low-level actions stored in the file referenced by ID 001.

```
PlayJrnl "001"
```

**See Also** None.

Pmt

## Pmt Function

---

**Description** Returns a constant periodic payment amount for an annuity or a loan.

**Syntax** **Pmt** (*rate*, *nper*, *pv*, *fv*, *due*)

| Syntax Element | Description                                                                                                                 |
|----------------|-----------------------------------------------------------------------------------------------------------------------------|
| <i>rate</i>    | Interest rate per period.                                                                                                   |
| <i>nper</i>    | Total number of payment periods.                                                                                            |
| <i>pv</i>      | Present value of the initial lump sum amount paid (as in the case of an annuity) or received (as in the case of a loan).    |
| <i>fv</i>      | Future value of the final lump sum amount required (as in the case of a savings plan) or paid (0 as in the case of a loan). |
| <i>due</i>     | An integer value for when the payments are due (0=end of each period, 1= beginning of the period).                          |

**Comments** *Rate* is assumed to be constant over the life of the loan or annuity. If payments are on a monthly schedule, then *rate* will be 0.0075 if the annual percentage rate on the annuity or loan is 9%.

**Example** This example finds the monthly payment on a given loan.

```
Sub main
 Dim aprate, totalpay
 Dim loanpv, loanfv
 Dim due, monthlpay
 Dim yearlpay, msgtext
 loanpv=InputBox("Enter the loan amount: ")
 aprate=InputBox("Enter the loan rate percent: ")
 If aprate > 1 then
 Aprate = aprate/100
 End If
 totalpay=InputBox("Enter the total number of monthly payments: ")
 loanfv=0
 'Assume payments are made at end of month
 due=0
 monthlpay=Pmt(aprate/12,totalpay,-loanpv,loanfv,due)
 msgtext="The monthly payment is: " & Format(monthlpay,"Currency")
 MsgBox msgtext
End Sub
```

**See Also**

|      |      |
|------|------|
| FV   | PV   |
| IPmt | PPmt |
| IRR  | Rate |
| NPV  |      |

# PopupMenuIDSelect

User Action Command

»»»SQA»

**Description** Performs a popup menu selection based on the internal ID of the menu item.

**Syntax** **PopupMenuIDSelect** *MenuID&*

| Syntax Element     | Description                       |
|--------------------|-----------------------------------|
| <i>MenuID&amp;</i> | The internal ID of the menu item. |

**Comments** This command is usually preceded by a command containing a mouse-click action required to activate the popup menu.

This command is necessary for making selections from popup menu items that do not contain text, such as owner drawn or bitmap menus.

**Example** This example clicks the right mouse button at the *x,y* coordinates of 50,43 in the current context window and then selects the menu item identified by the internal ID 1145 from the pop-up menu that appears.

```
Window Right_Click, "", "Coords=50,43"
PopupMenuIDSelect 1145
```

**See Also** MenuIDSelect            SysMenuIDSelect  
 MenuSelect                SysMenuSelect  
 PopupMenuSelect

# PopupMenuSelect

User Action Command

»»»SQA»

**Description** Selects a popup menu item through one or more mouse clicks.

**Syntax** **PopupMenuSelect** *menuPath\$*

| Syntax Element    | Description                                                                                                                                                                                                                                                                                                                                                                   |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>menuPath\$</i> | A sequential list of the popup menu's sub-menus, if any, and the target menu item that a user clicks. Each is separated by a pointer (->).<br><br>If you are specifying an item by position or by ID rather than by name, <i>menuPath</i> must begin with Menu=. For example, Menu=pos(3) selects the third item in the popup menu. See <i>Comments</i> for more information. |

## PopupMenuSelect

**Comments** This command is usually preceded by a command containing a mouse-click action required to activate the popup menu.

During recording, Robot identifies menu item selections by item name. Each name represents a mouse click. For example, Robot might record a command to add a new account to a database as follows:

```
PopupMenuSelect "Add Account..." User clicks Add Account
```

During manual scripting, you can reference a popup menu item selection in any of the following ways:

- ▶ Through the menu item name:

```
PopupMenuSelect "Add Account..."
```

- ▶ Through the position of the menu item on the menu:

```
PopupMenuSelect "menu=pos(3)"
```

The first item in a menu is position 1, not 0. Also, ignore menu item separators when counting the position of an item in a menu.

- ▶ Through the menu item ID:

```
PopupMenuSelect "menu=id(9270)"
```

You can use any of the above methods to represent both intermediate menu items and the target menu item.

When using `PopupMenuSelect` to select a menu item, you must reference every sub-menu, if any, up to and including the menu where the target item is located. However, you can select a menu item directly by its item ID, without specifying any sub-menu, by calling `PopupMenuIDSelect`.

During manual scripting, you can select a popup menu item through a series of `InputKeys` commands, or through a combination of `PopupMenuSelect` and `InputKeys` commands. This feature lets you play back a menu item selection entirely through keystrokes, or through a combination of keystrokes and mouse clicks, rather than through mouse clicks alone. For example, the following commands select the menu item `Folder` from the `Windows Desktop` popup menu and `New` sub-menu:

```
Window SetContext, "Caption=Program Manager", ""
ListView Right_Click, "ObjectIndex=1", "Coords=27,966"
PopupMenuSelect "New" ' PopupMenuSelect "menu=pos(6)" also works
InputKeys "f"
```

If a popup menu is displayed, you can clear it by calling `PopupMenuSelect ""`.

**Example** This example clicks the right mouse button at the *x,y* coordinates of 50,43 in the current context window and then select the menu item Attributes... from the pop-up menu that appears.

```
Window Right_Click, "", "Coords=50,43"
PopupMenuSelect "Attributes..."
```

**See Also**

|                   |                 |
|-------------------|-----------------|
| MenuIDSelect      | SysMenuIDSelect |
| MenuSelect        | SysMenuSelect   |
| PopupMenuIDSelect |                 |

## PPmt

### Function

---

**Description** Returns the principal portion of the payment for a given period of an annuity.

**Syntax** **PPmt** (*rate*, *per*, *nper*, *pv*, *fv*, *due*)

| Syntax Element | Description                                                                                                                 |
|----------------|-----------------------------------------------------------------------------------------------------------------------------|
| <i>rate</i>    | Interest rate per period.                                                                                                   |
| <i>per</i>     | Particular payment period in the range 1 through <i>nper</i> .                                                              |
| <i>nper</i>    | Total number of payment periods.                                                                                            |
| <i>pv</i>      | Present value of the initial lump sum amount paid (as in the case of an annuity) or received (as in the case of a loan).    |
| <i>fv</i>      | Future value of the final lump sum amount required (as in the case of a savings plan) or paid (0 as in the case of a loan). |
| <i>due</i>     | An integer value for when the payments are due (0=end of each period, 1= beginning of the period).                          |

**Comments** *Rate* is assumed to be constant over the life of the loan or annuity. If payments are on a monthly schedule, then *rate* will be 0.0075 if the annual percentage rate on the annuity or loan is 9%.

Print

**Example** This example finds the principal portion of a loan payment amount for payments made in last month of the first year. The loan is for \$25,000 to be paid back over 5 years at 9.5% interest.

```
Sub main
 Dim aprate, periods
 Dim payperiod
 Dim loanpv, due
 Dim loanfv, principal
 Dim msgtext
 aprate=9.5/100
 payperiod=12
 periods=120
 loanpv=25000
 loanfv=0
 Rem Assume payments are made at end of month
 due=0
 principal=PPmt(aprate/12,payperiod,periods,- loanpv,loanfv,due)
 msgtext="Given a loan of $25,000 @ 9.5% for 10 years," & Chr(10)
 msgtext=msgtext & " the principal paid in month 12 is: "
 MsgBox msgtext & Format(principal, "Currency")
End Sub
```

**See Also**

|      |      |
|------|------|
| FV   | Pmt  |
| IPmt | PV   |
| IRR  | Rate |
| NPV  |      |

## Print Statement

---

**Description** Prints data to an open file or to the screen.

**Syntax** **Print** [[#*filenumber%*,] *expressionlist* [{;|,}]]

| Syntax Element        | Description                                                                                                            |
|-----------------------|------------------------------------------------------------------------------------------------------------------------|
| # <i>filenumber%</i>  | An integer expression identifying the open file to write to. The pound sign (#) preceding the file number is required. |
| <i>expressionlist</i> | A numeric, string, and Variant expression containing the list of values to print.                                      |

**Comments** The `Print` statement outputs data to the specified *filenumber%*. *filenumber%* is the number assigned to the file when it was opened. See the `Open` statement for more information. If this argument is omitted, the `Print` statement outputs data to the screen.

If the *expressionlist* is omitted, a blank line is written to the file.



The values in *expressionlist* are separated by either a semicolon (;) or a comma (,). A semicolon indicates that the next value should appear immediately after the preceding one without intervening white space. A comma indicates that the next value should be positioned at the next print zone. Print zones begin every 14 spaces.

The optional [*{;|,}*] argument at the end of the `Print` statement determines where output for the next `Print` statement to the same output file should begin. A semicolon will place output immediately after the output from this `Print` statement on the current line; a comma will start output at the next print zone on the current line. If neither separator is specified, a CR-LF pair will be generated and the next `Print` statement will print to the next line.

Special functions `Spc` and `Tab` can be used inside `Print` statement to insert a given number of spaces and to move the print position to a desired column.

The `Print` statement supports only elementary SQABasic data types. See `Input` for more information on parsing this statement.

**Example** This example prints to the screen the octal values for the numbers 1 through 25.

```
Sub Main
 Dim x as Integer
 Dim y
 For x=1 to 25
 y=Oct$(x)
 Print x Tab(10) y
 Next x
End Sub
```

This example prints the string *myString* to the file *sFilename*.

```
Sub Main
 Dim myString as String
 Dim sFilename as String
 myString = "ABCDEFGHJIJ0123456789"
 sFilename = "C:\Temp0001.txt"
 Open sFilename For Output As #1
 Print #1, myString
 Close #1
End Sub
```

**See Also** `Open`      `Tab`  
`Spc`            `Write`

**Private**  
Keyword

»»SQA»

---

`Private` is an unused reserved keyword.

# ProgressBar

User Action Command

**Description** Performs an action on a progress bar control.

**Syntax** **ProgressBar** *action%*, *recMethod\$*, *parameters\$*

| Syntax Element     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i>     | <p>One of these mouse actions:</p> <ul style="list-style-type: none"> <li>▶ <i>MouseClicked</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> must contain <i>Coords=x,y</i>.</li> <li>▶ <i>MouseDown</i>. The dragging of the mouse while mouse buttons and/or shifting keys (Ctrl, Alt, Shift) are pressed. When <i>action%</i> contains a mouse-drag value, <i>parameters\$</i> must contain <i>Coords=x1,y1,x2,y2</i>.</li> </ul> <p>See Appendix E for a list of mouse click and drag values.</p>                                                                                                                                                                                                                                                                                                                                                                                                 |
| <i>recMethod\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <i>ID=%</i>. The object's internal Windows ID.</li> <li>▶ <i>Name=\$</i>. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be <i>Command1</i>.</li> <li>▶ <i>ObjectIndex=%</i>. The number of the object among all objects of the same type in the same window.</li> <li>▶ <i>State=\$</i>. An optional qualifier for any other recognition method. There are two possible values for this setting: <i>Enabled</i> and <i>Disabled</i>. The default state is the state of the current context window (as set in the most recent <i>Window SetContext</i> command), or <i>Enabled</i> if the state has not been otherwise declared.</li> <li>▶ <i>Text=\$</i>. The text displayed on the object.</li> <li>▶ <i>VisualText=\$</i>. An optional setting used to identify an object by its prior label. It is for user clarification only and does not affect object recognition.</li> </ul> |

▶ ▶ ▶

▶ ▶ ▶

| Syntax Element      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>parameters\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <i>Coords=x,y</i>. If <i>action%</i> is a mouse click, specifies the coordinates of the click, relative to the top left of the object.</li> <li>▶ <i>Coords=x1,y1,x2,y2</i>. If <i>action%</i> is a mouse drag, specifies the coordinates, where <i>x1,y1</i> are the starting coordinates of the drag, and <i>x2,y2</i> are the ending coordinates. The coordinates are relative to the top left of the object.</li> </ul> |

Comments None.

Example This example clicks the first progress bar control in the window (ObjectIndex=1) at *x,y* coordinates of 50,25.

```
ProgressBar Click, "ObjectIndex=1", "Coords=50,25"
```

See Also ProgressBarVP

## ProgressBarVP

Verification Point Command

▶▶SQA▶

Description Establishes a verification point for a progress bar control.

Syntax *Result* = **ProgressBarVP** (*action%*, *recMethod\$*, *parameters\$*)

| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i> | <p>The type of verification to perform. Valid values:</p> <ul style="list-style-type: none"> <li>▶ <i>CompareNumeric</i>. Captures the numeric value of the text of the object and compares it to the value of <i>parameters\$ Value or Range</i>. <i>parameters\$ VP</i> and either <i>Value or Range</i> are required; <i>ExpectedResult</i> and <i>Wait</i> are optional.</li> <li>▶ <i>CompareProperties</i>. Captures object properties information for the object and compares it to a recorded baseline. <i>parameters\$ VP</i> is required; <i>ExpectedResult</i> and <i>Wait</i> are optional.</li> <li>▶ <i>CompareText</i>. Captures the text of the object and compares it to a recorded baseline. <i>parameters\$ VP</i> and <i>Type</i> are required; <i>ExpectedResult</i> and <i>Wait</i> are optional.</li> </ul> |

▶ ▶ ▶



| Syntax Element       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>recMethod</i> \$  | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ ID=% . The object's internal Windows ID.</li> <li>▶ Name=\$ . A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be Command1.</li> <li>▶ ObjectIndex=% . The number of the object among all objects of the same type in the same window.</li> <li>▶ Text=\$ . The text displayed on the object.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <i>parameters</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ ExpectedResult=% . Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> <li>- PASS. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail.</li> <li>- FAIL. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail.</li> </ul> </li> <li>▶ Range=&amp; , &amp; . Used with the action <code>CompareNumeric</code> when a numeric range comparison is being performed, as in <code>Range=2 , 12</code> (test for numbers in this range). The values are inclusive.</li> <li>▶ Type=\$ . Specifies the verification method to use for <code>CompareText</code> actions. The possible values are: <code>CaseSensitive</code>, <code>CaseInsensitive</code>, <code>FindSubStr</code>, <code>FindSubStrI</code> (case insensitive), and <code>UserDefined</code>. See <i>Comments</i> for more information. If <code>UserDefined</code> is specified, two additional parameters are required: <ul style="list-style-type: none"> <li>- DLL=\$ . The full path and file name of the library that contains the function</li> <li>- Function=\$ . The name of the custom function to use in comparing the text</li> </ul> </li> <li>▶ Value=&amp; . Used with the action <code>CompareNumeric</code> when a numeric equivalence comparison is being performed, as in <code>Value=25</code> (test against the value 25).</li> <li>▶ VP=\$ . The verification point ID. IDs must be unique within a script. Required for all verification points.</li> <li>▶ Wait=% , % . A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10 , 40</code> (retry the test every 10 seconds, but time out the test after 40 seconds).</li> </ul> |

**Comments** This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

With the `Type=$` parameter, `CaseSensitive` and `CaseInsensitive` require a full match between the current baseline text and the text captured during playback. With `FindSubStr` and `FindSubStrI`, the current baseline can be a substring of the text captured during playback. The substring can appear anywhere in the playback text. To modify the current baseline text, double-click the verification point name in the Robot Asset pane (to the left of the script).

**Example** This example captures the properties of the first progress bar control in the window (`ObjectIndex=1`) and compares them to the recorded baseline in verification point `TEST1A`.

```
Result = ProgressBarVP (CompareProperties, "ObjectIndex=1",
"VP=TEST1A")
```

**See Also** `ProgressBar`

## PSCalendar

User Action Command

»»SQA»

---

This command is obsolete and should not be used. It continues to be supported to maintain the upward compatibility of your existing scripts.

## PSCalendarVP

Verification Point Command

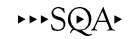
»»SQA»

---

This command is obsolete and should not be used. It continues to be supported to maintain the upward compatibility of your existing scripts.

## PSGrid

User Action Command



**Description** Performs an action on a PeopleTools grid.

**Syntax** **PSGrid** *action%*, *recMethod\$*, *parameters\$*

| Syntax Element     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i>     | <p>One of these actions:</p> <ul style="list-style-type: none"> <li>▶ <i>MouseClicked</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift).</li> <li>▶ When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> identifies the grid <i>row</i> that was clicked. See <i>Comments</i> for more information.</li> <li>▶ <i>MouseDown</i>. The dragging of the mouse while mouse buttons and/or shifting keys (Ctrl, Alt, Shift) are pressed. When <i>action%</i> contains a mouse-drag value, <i>parameters\$</i> must contain <i>Coords=x1,y1,x2,y2</i>. See Appendix E for a list of mouse click and drag values.</li> <li>▶ <i>ScrollAction</i>. One of these scroll actions: <ul style="list-style-type: none"> <li>ScrollPageRight      ScrollPageDown</li> <li>ScrollRight          ScrollLineDown</li> <li>ScrollPageLeft      ScrollPageUp</li> <li>ScrollLeft            ScrollLineUp</li> <li>HScrollTo            VScrollTo</li> </ul> </li> </ul> <p>HScrollTo and VScrollTo take the required parameter <i>Position=%</i>.</p> |
| <i>recMethod\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <i>ColIndex=%</i>. In grids that let you select individual cells (such as the Data Designer), a zero based value that identifies the column that was clicked. Used only after one of these parent values: <i>ID=%</i>, <i>ObjectIndex=%</i>, <i>Text=\$</i>. Parent/child values are separated by a backslash and semicolons (<i>i\;</i>).</li> <li>▶ <i>Heading=\$</i>. In grids that let you select individual cells (such as the Data Designer), identifies the column that was clicked. Used only after one of these parent values: <i>ID=%</i>, <i>ObjectIndex=%</i>, <i>Text=\$</i>. Parent/child values are separated by a backslash and semicolons (<i>i\;</i>).</li> <li>▶ <i>ID=%</i>. The object's internal Windows ID.</li> </ul>                                                                                                                                                                                                                                                                                                                                      |



▶ ▶ ▶

| Syntax Element      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                     | <ul style="list-style-type: none"> <li>▶ <code>ObjectIndex=%</code>. The number of the object among all objects of the same type in the same window.</li> <li>▶ <code>State=\$</code>. An optional qualifier for any other recognition method. There are two possible values for this setting: <code>Enabled</code> and <code>Disabled</code>. The default state is the state of the current context window (as set in the most recent <code>Window SetContext</code> command), or <code>Enabled</code> if the state has not been otherwise declared.</li> <li>▶ <code>Text=\$</code>. The text displayed on the object.</li> <li>▶ <code>VisualText=\$</code>. An optional setting used to identify an object by its prior label. It is for user clarification only and does not affect object recognition.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <i>parameters\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>Col=%;Value=x</code>. If <i>action%</i> is a mouse click, these two parameters specify the row that was clicked: <ul style="list-style-type: none"> <li>- <code>Col</code> is the numeric position of a column in the grid (the leftmost column = 1, the next column = 2, etc.)</li> <li>- <code>Value</code> is the contents of the cell located at the intersection of column <code>Col</code> and the clicked row</li> </ul> </li> <li>▶ <code>ColTitle=\$;Value=x</code>. If <i>action%</i> is a mouse click, these two parameters specify the row that was clicked: <ul style="list-style-type: none"> <li>- <code>ColTitle</code> is a column heading</li> <li>- <code>Value</code> is the contents of the cell located at the intersection of the column with the heading <code>ColTitle</code> and the clicked row</li> </ul> </li> <li>▶ <code>Coords=x,y</code>. If <i>action%</i> is a mouse click, specifies the <i>x,y</i> coordinates of the click, relative to the top left of the clicked cell or column.</li> <li>▶ <code>Coords=x1,y1,x2,y2</code>. If <i>action%</i> is a mouse drag, specifies the coordinates, where <i>x1,y1</i> are the starting coordinates of the drag, and <i>x2,y2</i> are the ending coordinates. The coordinates are relative to the top left of the object or the item.</li> <li>▶ <code>Position=%</code>. If <i>action%</i> is <code>VScrollTo</code> or <code>HScrollTo</code>, specifies the scroll bar value of the new scrolled-to position in the scroll box. Every scroll bar has an internal range, and this value is specific to that range.</li> <li>▶ <code>Row=%</code>. If <i>action%</i> is a mouse click, the number of the row that was clicked (the topmost row = 1).</li> <li>▶ <code>Text=\$</code>. If <i>action%</i> is a mouse click, the visible text in the row that was clicked.</li> </ul> |

## PSGrid

- Comments With mouse-click actions, *recMethod*\$ may specify the column that was clicked, and *parameters*\$ may specify the row that was clicked.
- Robot specifies the clicked row by using one of these *parameters*\$ values (or pairs of values):
- ▶ One or more pairs of a column identifier (*Col*=% or *ColTitle*=%) followed by *Value*=x. Robot uses as many column/value pairs as necessary to uniquely identify the clicked row — for example:  

```
"ColTitle=Cntry;Value=USA;ColTitle=St;Value=AR;Col=3;Value=18"
```
  - ▶ *Text*=\$. Text values from multiple columns are separated with a pipe separator ( | ) — for example:  

```
"Text=9|0|Edit|Drop Down List|AE_MENU_EDIT|AE_WRK"
```
  - ▶ Optionally, you can use the tab separator *Chr*\$(9) instead of the pipe separator.
  - ▶ *Row*=%.
  - ▶ *Coords*=x,y.

Note the following points about column/value pairs:

- ▶ Value must *immediately* follow *Col* or *ColTitle*.
- ▶ The values are separated by a semicolon ( ;) — for example:  

```
"ColTitle=Customer ID;Value=0253319"
```
- ▶ The column identifier (*Col* or *ColTitle*) isn't necessarily the column that was clicked. Robot looks for one or more columns of unique values. If a key column is found:
  - The column identifier specifies the key column
  - Value specifies the contents of the cell at the intersection of the key column and the row that the user clicked

*parameters*\$ has a maximum length of 968 characters. If multiple column/row pairs cause *parameters*\$ to exceed the maximum length, Robot uses another way to uniquely identify the clicked row.

- Example In this example, a PeopleSoft grid is clicked. The grid is identified as object 1 in the current context window. The column that was clicked is identified by the heading Count.

```
PSGrid Click, "ObjectIndex=1",Text=6| 0|Message Underline|Frame||"
```

## See Also

|                |               |                |
|----------------|---------------|----------------|
| PSGridHeader   | PSNavigator   | PSSpinVP       |
| PSGridHeaderVP | PSNavigatorVP | PSTree         |
| PSGridVP       | PSPanel       | PSTreeHeader   |
| PSMenu         | PSPanelVP     | PSTreeHeaderVP |
| PSMenuVP       | PSSpin        | PSTreeVP       |



# PSGridHeader

User Action Command

»»SQA»

**Description** Performs an action on a column header in a PeopleTools grid.

**Syntax** **PSGridHeader** *action%*, *recMethod\$*, *parameters\$*

| Syntax Element      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i>      | <p>One of these mouse actions:</p> <ul style="list-style-type: none"> <li>▶ <i>MouseClicked</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> must contain <i>Coords=x,y</i>.</li> <li>▶ <i>MouseDown</i>. The dragging of the mouse while mouse buttons and/or shifting keys (Ctrl, Alt, Shift) are pressed. When <i>action%</i> contains a mouse-drag value, <i>parameters\$</i> must contain <i>Coords=x1,y1,x2,y2</i>.</li> </ul> <p>See Appendix E for a list of mouse click and drag values.</p>                                                                                                                                                                              |
| <i>recMethod\$</i>  | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <i>ID=%</i>. The object's internal Windows ID.</li> <li>▶ <i>ObjectIndex=%</i>. The number of the object among all objects of the same type in the same window.</li> <li>▶ <i>State=\$</i>. An optional qualifier for any other recognition method. There are two possible values for this setting: <i>Enabled</i> and <i>Disabled</i>. The default state is the state of the current context window (as set in the most recent <i>Window SetContext</i> command), or <i>Enabled</i> if the state has not been otherwise declared.</li> <li>▶ <i>Text=\$</i>. The text displayed on the object.</li> <li>▶ <i>VisualText=\$</i>. An optional setting used to identify an object by its visible text. It is for user clarification only and does not affect object recognition.</li> </ul> |
| <i>parameters\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <i>Coords=x,y</i>. If <i>action%</i> is a mouse click, specifies the <i>x,y</i> coordinates of the click, relative to the top left of the object or the item.</li> <li>▶ <i>Coords=x1,y1,x2,y2</i>. If <i>action%</i> is a mouse drag, specifies the coordinates, where <i>x1,y1</i> are the starting coordinates of the drag, and <i>x2,y2</i> are the ending coordinates. The coordinates are relative to the top left of the object or the item.</li> </ul>                                                                                                                                                                                                                                                                                                                            |

PSGridHeaderVP

Comments Robot only supports actions against visible headers.

Example In this example, a PeopleSoft grid header is clicked. The grid is identified as object 1 in the current context window.

```
PSGridHeader Click, "ObjectIndex=1", "Coords=328,4"
```

See Also

|                |               |                |
|----------------|---------------|----------------|
| PSGrid         | PSNavigator   | PSSpinVP       |
| PSGridHeaderVP | PSNavigatorVP | PSTree         |
| PSGridVP       | PSPanel       | PSTreeHeader   |
| PSMenu         | PSPanelVP     | PSTreeHeaderVP |
| PSMenuVP       | PSSpin        | PSTreeVP       |

## PSGridHeaderVP

Verification Point Command



Description Establishes a verification point for a column header in a PeopleTools grid.

Syntax *Result* = **PSGridHeaderVP** (*action%*,*recMethod*,\$*parameters*\$)

| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i> | <p>The type of verification to perform. Valid values:</p> <ul style="list-style-type: none"><li>▶ <b>CompareData</b>. Captures the data of the object and compares it to a recorded baseline. <i>parameters</i>\$ <i>VP</i> and <i>Type</i> are required; <i>ExpectedResult</i> and <i>Wait</i> are optional.</li><li>▶ <b>CompareNumeric</b>. Captures the numeric value of the text of the object and compares it to the value of <i>parameters</i>\$ <i>Value</i> or <i>Range</i>. <i>parameters</i>\$ <i>VP</i> and either <i>Value</i> or <i>Range</i> are required; <i>ExpectedResult</i> and <i>Wait</i> are optional.</li><li>▶ <b>CompareProperties</b>. Captures object properties information for the object and compares it to a recorded baseline. <i>parameters</i>\$ <i>VP</i> is required; <i>ExpectedResult</i> and <i>Wait</i> are optional.</li><li>▶ <b>CompareText</b>. Captures the text of the object and compares it to a recorded baseline. <i>parameters</i>\$ <i>VP</i> and <i>Type</i> are required; <i>ExpectedResult</i> and <i>Wait</i> are optional.</li></ul> |



▶ ▶ ▶

| Syntax Element       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>recMethod</i> \$  | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ ID=%. The object's internal Windows ID.</li> <li>▶ ObjectIndex=%. The number of the object among all objects of the same type in the same window.</li> <li>▶ Text=\$. The text displayed on the object.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <i>parameters</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ ExpectedResult=%. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> <li>- PASS. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail.</li> <li>- FAIL. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail.</li> </ul> </li> <li>▶ Range=&amp; , &amp;. Used with the action <code>CompareNumeric</code> when a numeric range comparison is being performed, as in <code>Range=2,12</code> (test for numbers in this range). The values are inclusive.</li> <li>▶ Type=\$. Specifies the verification method to use for <code>CompareText</code> actions. The possible values are: <code>CaseSensitive</code>, <code>CaseInsensitive</code>, <code>FindSubStr</code>, <code>FindSubStrI</code> (case insensitive), and <code>UserDefined</code>. See <i>Comments</i> for more information. If <code>UserDefined</code> is specified, two additional parameters are required: <ul style="list-style-type: none"> <li>- DLL=\$. The full path and file name of the library that contains the function</li> <li>- Function=\$. The name of the custom function to use in comparing the text</li> </ul> </li> <li>▶ Value=&amp;. Used with the action <code>CompareNumeric</code> when a numeric equivalence comparison is being performed, as in <code>Value=25</code> (test against the value 25).</li> <li>▶ VP=\$. The verification point ID. IDs must be unique within a script. Required for all verification points.</li> <li>▶ Wait=% , %. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10,40</code> (retry the test every 10 seconds, but time out the test after 40 seconds).</li> </ul> |

## PSGridVP

**Comments** This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

With the `Type=$` parameter, `CaseSensitive` and `CaseInsensitive` require a full match between the current baseline text and the text captured during playback. With `FindSubStr` and `FindSubStrI`, the current baseline can be a substring of the text captured during playback. The substring can appear anywhere in the playback text. To modify the current baseline text, double-click the verification point name in the Robot Asset pane (to the left of the script).

Robot only supports the testing of visible headers.

**Example** In this example, an object data verification point is established for a PeopleSoft grid header. The grid is identified as object 1 in the current context window.

```
Result = PSGridHeaderVP (CompareData, "ObjectIndex=1", "VP=GRDTST")
```

### See Also

|              |               |                |
|--------------|---------------|----------------|
| PSGrid       | PSNavigator   | PSSpinVP       |
| PSGridHeader | PSNavigatorVP | PSTree         |
| PSGridVP     | PSPanel       | PSTreeHeader   |
| PSMenu       | PSPanelVP     | PSTreeHeaderVP |
| PSMenuVP     | PSSpin        | PSTreeVP       |

## PSGridVP

### Verification Point Command



**Description** Establishes a verification point for a PeopleTools grid.

**Syntax** `Result = PSGridVP (action%, recMethod$, parameters$)`

| Syntax Element       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>action%</code> | <p>The type of verification to perform. Valid values:</p> <ul style="list-style-type: none"><li>▶ <code>CompareData</code>. Captures the data of the object and compares it to a recorded baseline. <code>parameters\$ VP</code> and <code>Type</code> are required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li><li>▶ <code>CompareNumeric</code>. Captures the numeric value of the text of the object and compares it to the value of <code>parameters\$ Value</code> or <code>Range</code>. <code>parameters\$ VP</code> and either <code>Value</code> or <code>Range</code> are required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li></ul> |



▶ ▶ ▶

| Syntax Element      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                     | <ul style="list-style-type: none"> <li>▶ <code>CompareProperties</code>. Captures object properties information for the object and compares it to a recorded baseline. <i>parameters\$</i> VP is required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li> <li>▶ <code>CompareText</code>. Captures the text of the selected row and compares it to a recorded baseline. <i>parameters\$</i> VP and <code>Type</code> are required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <i>recMethod\$</i>  | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>ColIndex=%</code>. In grids that let you select individual cells (such as the Data Designer), a zero based value that identifies the column that was clicked. Used only after one of these parent values: <code>ID=%</code>, <code>ObjectIndex=%</code>, <code>Text=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>i\;</code>).</li> <li>▶ <code>Heading=\$</code>. In grids that let you select individual cells (such as the Data Designer), identifies the column that was clicked. Used only after one of these parent values: <code>ID=%</code>, <code>ObjectIndex=%</code>, <code>Text=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>i\;</code>).</li> <li>▶ <code>ID=%</code>. The object's internal Windows ID.</li> <li>▶ <code>ObjectIndex=%</code>. The number of the object among all objects of the same type in the same window.</li> <li>▶ <code>Text=\$</code>. The text displayed on the object.</li> </ul> |
| <i>parameters\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>ExpectedResult=%</code>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> <li>- <code>PASS</code>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail.</li> <li>- <code>FAIL</code>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail.</li> </ul> </li> <li>▶ <code>Range=&amp; , &amp;</code>. Used with the action <code>CompareNumeric</code> when a numeric range comparison is being performed, as in <code>Range=2 , 12</code> (test for numbers in this range). The values are inclusive.</li> </ul>                                                                                                                                  |

▶ ▶ ▶



| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                | <ul style="list-style-type: none"> <li>▶ <b>Type=\$</b>. Specifies the verification method to use for CompareText actions. The possible values are: CaseSensitive, CaseInsensitive, FindSubStr, FindSubStrI (case insensitive), and UserDefined. See <i>Comments</i> for more information. If UserDefined is specified, two additional parameters are required:               <ul style="list-style-type: none"> <li>- <b>DLL=\$</b>. The full path and file name of the library that contains the function</li> <li>- <b>Function=\$</b>. The name of the custom function to use in comparing the text</li> </ul> </li> <li>▶ <b>Value=&amp;</b>. Used with the action CompareNumeric when a numeric equivalence comparison is being performed, as in Value=25 (test against the value 25).</li> <li>▶ <b>VP=\$</b>. The verification point ID. IDs must be unique within a script. Required for all verification points.</li> <li>▶ <b>Wait=% , %</b>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in Wait=10 , 40 (retry the test every 10 seconds, but time out the test after 40 seconds).</li> </ul> |

**Comments** This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

With the Type=\$ parameter, CaseSensitive and CaseInsensitive require a full match between the current baseline text and the text captured during playback. With FindSubStr and FindSubStrI, the current baseline can be a substring of the text captured during playback. The substring can appear anywhere in the playback text. To modify the current baseline text, double-click the verification point name in the Robot Asset pane (to the left of the script).

Robot supports the testing of all columns in the grid, whether or not a column is visible.

**Example** In this example, an object properties verification point is established for a PeopleSoft grid. The grid is identified as object 1 in the current context window.

```
Result = PSGridVP(CompareProperties, "ObjectIndex=1", "VP=GRDPRPS")
```

**See Also**

|                |               |                |
|----------------|---------------|----------------|
| PSGrid         | PSNavigator   | PSSpinVP       |
| PSGridHeader   | PSNavigatorVP | PSTree         |
| PSGridHeaderVP | PSPanel       | PSTreeHeader   |
| PSMenu         | PSPanelVP     | PSTreeHeaderVP |
| PSMenuVP       | PSSpin        | PSTreeVP       |

# PSMenu

## User Action Command



**Description** Performs an action on a PeopleTools menu object.

**Syntax** **PSMenu** *action%*, *recMethod\$*, *parameters\$*

| Syntax Element      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i>      | <p>The following mouse action:</p> <ul style="list-style-type: none"> <li>▶ <i>MouseClicked</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). Does not require coordinate information.</li> </ul> <p>See Appendix E for a list of mouse click values.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <i>recMethod\$</i>  | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <i>ObjectIndex=%</i>. The number of the object among all objects of the same type (PSMenu) in the same window.</li> <li>▶ If the action occurs on a menu item, <i>recMethod\$</i> includes one or more of these child values: <ul style="list-style-type: none"> <li>- <i>Class=\$</i>. The class name of the menu item.</li> <li>- <i>ClassIndex=%</i>. Index of the object with the same class name value.</li> <li>- <i>Index=%</i>. Index of the menu item acted upon.</li> <li>- <i>Name=\$</i>. Object name of the menu item.</li> <li>- <i>Type=\$</i>. Either <i>PSMenuBarItem</i> or <i>PSMenuItem</i>.</li> </ul> </li> <li>▶ These child values are used only after the parent value <i>ObjectIndex=%</i>. Parent/child values are separated by a backslash and semicolons (<i>;\;</i>).</li> </ul> |
| <i>parameters\$</i> | <p>Valid value:</p> <ul style="list-style-type: none"> <li>▶ <i>Coords=x,y</i>. Specifies the <i>x,y</i> coordinates of the click, relative to the top left of the object or the item.</li> <li>▶ If no coordinates are specified, the coordinates at the center of the object or item are used.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |

**Comments** None.

**Example** In this example, the user clicks a menu item in a PeopleTools design window. The object name of the menu item is MENUITEM1.

```
Window SetContext, "Caption=DATA_DESIGNER (MENU);Childwindow", ""
PSMenu Click, "ObjectIndex=1;\;Name=MENUITEM1, "Coords=2,8"
```

## PSMenuVP

|          |                |               |                |
|----------|----------------|---------------|----------------|
| See Also | PSGrid         | PSNavigator   | PSSpinVP       |
|          | PSGridHeader   | PSNavigatorVP | PSTree         |
|          | PSGridHeaderVP | PSPanel       | PSTreeHeader   |
|          | PSGridVP       | PSPanelVP     | PSTreeHeaderVP |
|          | PSMenuVP       | PSSpin        | PSTreeVP       |

## PSMenuVP

Verification Point Command



**Description** Establishes a verification point for a PeopleTools menu object.

**Syntax** *Result* = **PSMenuVP** (*action%*, *recMethod\$*, *parameters\$*)

| Syntax Element      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i>      | The type of verification to perform. Valid value: <ul style="list-style-type: none"><li>▶ Compare. Captures the properties of the menu objects and compares them to the recorded baseline. <i>parameters\$</i> VP is required; <i>ExpectedResult</i> is optional.</li></ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <i>recMethod\$</i>  | Valid value: <ul style="list-style-type: none"><li>▶ ObjectIndex=%. The number of the object among all objects of the same type in the same window.</li></ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <i>parameters\$</i> | Valid values: <ul style="list-style-type: none"><li>▶ ExpectedResult=%. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values:<ul style="list-style-type: none"><li>– PASS. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail.</li><li>– FAIL. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail.</li></ul></li><li>▶ VP=\$. The verification point ID. IDs must be unique within a script. Required for all verification points.</li></ul> |

**Comments** This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

**Example** This example verifies a PeopleTools menu object.

```
Result = PSMenuVP (Compare, "ObjectIndex=1", "VP=FILEMNEU")
```



|          |                |               |                |
|----------|----------------|---------------|----------------|
| See Also | PSGrid         | PSNavigator   | PSSpinVP       |
|          | PSGridHeader   | PSNavigatorVP | PSTree         |
|          | PSGridHeaderVP | PSPanel       | PSTreeHeader   |
|          | PSGridVP       | PSPanelVP     | PSTreeHeaderVP |
|          | PSMenu         | PSSpin        | PSTreeVP       |

## PSNavigator

User Action Command

▶▶SQA▶

**Description** Performs an action on a PeopleTools Navigator window or a Navigator map in the PeopleTools Business Process Designer.

**Syntax** **PSNavigator** *action%*, *recMethod\$*, *parameters\$*

| Syntax Element     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                 |                |             |                |                |              |            |              |           |           |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|----------------|-------------|----------------|----------------|--------------|------------|--------------|-----------|-----------|
| <i>action%</i>     | <p>One of these actions:</p> <ul style="list-style-type: none"> <li>▶ <i>MouseClicked</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> must contain <i>Coords=x,y</i>.</li> <li>▶ <i>MouseDown</i>. The dragging of the mouse while mouse buttons and/or shifting keys (Ctrl, Alt, Shift) are pressed. When <i>action%</i> contains a mouse-drag value, <i>parameters\$</i> must contain <i>Coords=x1,y1,x2,y2</i>. See Appendix E for a list of mouse click and drag values.</li> <li>▶ <i>ScrollAction</i>. One of these scroll actions: <table style="margin-left: 20px; border: none;"> <tr> <td>ScrollPageRight</td> <td>ScrollPageDown</td> </tr> <tr> <td>ScrollRight</td> <td>ScrollLineDown</td> </tr> <tr> <td>ScrollPageLeft</td> <td>ScrollPageUp</td> </tr> <tr> <td>ScrollLeft</td> <td>ScrollLineUp</td> </tr> <tr> <td>HScrollTo</td> <td>VScrollTo</td> </tr> </table> <p>HScrollTo and VScrollTo take the required parameter <i>Position=%</i>.</p> </li> </ul> | ScrollPageRight | ScrollPageDown | ScrollRight | ScrollLineDown | ScrollPageLeft | ScrollPageUp | ScrollLeft | ScrollLineUp | HScrollTo | VScrollTo |
| ScrollPageRight    | ScrollPageDown                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                 |                |             |                |                |              |            |              |           |           |
| ScrollRight        | ScrollLineDown                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                 |                |             |                |                |              |            |              |           |           |
| ScrollPageLeft     | ScrollPageUp                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                 |                |             |                |                |              |            |              |           |           |
| ScrollLeft         | ScrollLineUp                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                 |                |             |                |                |              |            |              |           |           |
| HScrollTo          | VScrollTo                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |                 |                |             |                |                |              |            |              |           |           |
| <i>recMethod\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <i>ID=%</i>. The object's internal Windows ID.</li> <li>▶ <i>ItemID=%</i>. A zero-based value that identifies the map item that was clicked. Used only after one of these parent values: <i>ID=%</i>, <i>Name=\$</i>, <i>ObjectIndex=%</i>, <i>Text=\$</i>. Parent/child values are separated by a backslash and semicolons (<i>;\;</i>).</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                 |                |             |                |                |              |            |              |           |           |

▶ ▶ ▶



| Syntax Element      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                     | <ul style="list-style-type: none"> <li>▶ Name=\$. A name that a developer assigns to an object to uniquely identify the object in the development environment. Name can specify the Navigator object name or a map item name.</li> <li>▶ When Name specifies a map item, it is used only after one of these parent values: ID=%, Name=\$, ObjectIndex=%, Text=\$. Parent/child values are separated by a backslash and semicolons (; \ ;).</li> <li>▶ ObjectIndex=%. The number of the object among all objects of the same type in the same window.</li> <li>▶ State=\$. An optional qualifier for any other recognition method. There are two possible values for this setting: Enabled and Disabled. The default state is the state of the current context window (as set in the most recent Window SetContext command), or Enabled if the state has not been otherwise declared.</li> <li>▶ Text=\$. The text displayed on the object.</li> <li>▶ VisualText=\$. An optional setting used to identify an object by its visible text. It is for user clarification only and does not affect object recognition.</li> </ul> |
| <i>parameters\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ Coords=<i>x,y</i>. If <i>action%</i> is a mouse click, specifies the <i>x,y</i> coordinates of the click, relative to the top left of the object or the item.</li> <li>▶ Coords=<i>x1,y1,x2,y2</i>. If <i>action%</i> is a mouse drag, specifies the coordinates, where <i>x1,y1</i> are the starting coordinates of the drag, and <i>x2,y2</i> are the ending coordinates. The coordinates are relative to the top left of the object or the item.</li> <li>▶ Position=%. If <i>action%</i> is VScrollTo or HScrollTo, specifies the scroll bar value of the new scrolled-to position in the scroll box. Every scroll bar has an internal range, and this value is specific to that range.</li> </ul>                                                                                                                                                                                                                                                                                                                                                          |

Comments Robot recognizes a Navigator window by its internal map name.

Navigator windows are made up of items such as text, links, and steps. Robot considers all items inside a Navigator window to be map items. Different map items have different properties.

When you perform an action against a map item, Robot identifies the map item by an item ID or by its internal name as defined in the Business Process Designer.

For example, suppose you double click on a map item named Administrator Workflow in a Navigator window named PTDMO Default. Robot might define the *recMethod\$* argument as follows:

```
"Name=PTDMO Default;\;Name=Administrator Workflow"
```

Remember that the backslash character (\) indicates a parent-child relationship — in this case, between the Navigator window and a map item within the window.

**Example** This example clicks the PTDMO Default Navigator window.

```
PSNavigator Click, "Name=PTDMO Default", "Coords=10,5"
```

This example clicks the map item identified as item 0 in the Administer Workflow Navigator window.

```
PSNavigator Click, "Name=Administer Workflow;\;ItemID=0", ""
```

**See Also**

|                |               |                |
|----------------|---------------|----------------|
| PSGrid         | PSMenuVP      | PSSpinVP       |
| PSGridHeader   | PSNavigatorVP | PSTree         |
| PSGridHeaderVP | PSPanel       | PSTreeHeader   |
| PSGridVP       | PSPanelVP     | PSTreeHeaderVP |
| PSMenu         | PSSpin        | PSTreeVP       |

## PSNavigatorVP

Verification Point Command



**Description** Establishes a verification point for a PeopleTools Navigator window or a Navigator map in the PeopleTools Business Process Designer.

**Syntax** *Result* = **PSNavigatorVP**(*action%*,*recMethod\$*,*parameters\$*)

| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i> | <p>The type of verification to perform. Valid values:</p> <ul style="list-style-type: none"> <li>▶ <b>CompareData</b>. Captures the data of the object and compares it to a recorded baseline. <i>parameters\$ VP</i> and <b>Type</b> are required; <b>ExpectedResult</b> and <b>Wait</b> are optional.</li> <li>▶ <b>CompareNumeric</b>. Captures the numeric value of the text of the object and compares it to the value of <i>parameters\$ Value or Range</i>. <i>parameters\$ VP</i> and either <b>Value or Range</b> are required; <b>ExpectedResult</b> and <b>Wait</b> are optional.</li> </ul> |



▶ ▶ ▶

| Syntax Element      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                     | <ul style="list-style-type: none"> <li>▶ <code>CompareProperties</code>. Captures object properties information for the object and compares it to a recorded baseline. <i>parameters\$ VP</i> is required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li> <li>▶ <code>CompareText</code>. Captures the text of the object and compares it to a recorded baseline. <i>parameters\$ VP</i> and <code>Type</code> are required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <i>recMethod\$</i>  | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>ID=%</code>. The object's internal Windows ID.</li> <li>▶ <code>ItemID=%</code>. A zero-based value that identifies the map item that was clicked. Used only after one of these parent values: <code>ID=%</code>, <code>Name=\$</code>, <code>ObjectIndex=%</code>, <code>Text=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>;\</code>).</li> <li>▶ <code>Name=\$</code>. A name that a developer assigns to an object to uniquely identify the object in the development environment. <code>Name</code> can specify the Navigator object name or a map item name.</li> <li>▶ When <code>Name</code> specifies a map item, it is used only after one of these values: <code>ID=%</code>, <code>Name=\$</code>, <code>ObjectIndex=%</code>, <code>Text=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>;\</code>).</li> <li>▶ <code>ObjectIndex=%</code>. The number of the object among all objects of the same type in the same window.</li> <li>▶ <code>Text=\$</code>. The text displayed on the object.</li> </ul> |
| <i>parameters\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>ExpectedResult=%</code>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> <li>– <code>PASS</code>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail.</li> <li>– <code>FAIL</code>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail.</li> </ul> </li> <li>▶ <code>Range=&amp; , &amp;</code>. Used with the action <code>CompareNumeric</code> when a numeric range comparison is being performed, as in <code>Range=2 , 12</code> (test for numbers in this range). The values are inclusive.</li> </ul>                                                                                                                                                                                                                           |

▶ ▶ ▶



| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                | <ul style="list-style-type: none"> <li>▶ <code>Type=\$</code>. Specifies the verification method to use for <code>CompareText</code> actions. The possible values are: <code>CaseSensitive</code>, <code>CaseInsensitive</code>, <code>FindSubStr</code>, <code>FindSubStrI</code> (case insensitive), and <code>UserDefined</code>. See <i>Comments</i> for more information. If <code>UserDefined</code> is specified, two additional parameters are required: <ul style="list-style-type: none"> <li>- <code>DLL=\$</code>. The full path and file name of the library that contains the function</li> <li>- <code>Function=\$</code>. The name of the custom function to use in comparing the text</li> </ul> </li> <li>▶ <code>Value=&amp;</code>. Used with the action <code>CompareNumeric</code> when a numeric equivalence comparison is being performed, as in <code>Value=25</code> (test against the value 25).</li> <li>▶ <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points.</li> <li>▶ <code>Wait=% , %</code>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10 , 40</code> (retry the test every 10 seconds, but time out the test after 40 seconds).</li> </ul> |

**Comments** This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

With the `Type=$` parameter, `CaseSensitive` and `CaseInsensitive` require a full match between the current baseline text and the text captured during playback. With `FindSubStr` and `FindSubStrI`, the current baseline can be a substring of the text captured during playback. The substring can appear anywhere in the playback text. To modify the current baseline text, double-click the verification point name in the Robot Asset pane (to the left of the script).

Robot recognizes a Navigator window by its internal map name.

Navigator windows are made up of items such as text, links, and steps. Robot considers all items inside a Navigator window to be map items. Different map items have different properties.

When you verify a map item, Robot identifies the map item by an item ID or by an internal name as defined in the Business Process Designer. For example, suppose you establish a verification point for a map item named Administrator Workflow in a Navigator window named PTDMO Default. Robot might define the `recMethod$` argument as follows:

```
"Name=PTDMO Default;\;Name=Administrator Workflow"
```

## PSPanel

Remember that the backslash character (\) indicates a parent-child relationship — in this case, between the Navigator window and a map item within the window.

### Example

This example establishes an object properties verification point for the Navigator window Employee Training.

```
Result = PSNavigatorVP (CompareProperties,
 "Name=Employee Training", "VP=NAV")
```

This example establishes an object properties verification point for a map item. The map item is identified as object 4, which is in the Navigator object named Employee Training.

```
Result = PSNavigatorVP (CompareProperties,
 "Name=Employee Training;\;ItemID=4", "VP=NAVTRN")
```

### See Also

|                |             |                |
|----------------|-------------|----------------|
| PSGrid         | PSMenuVP    | PSSpinVP       |
| PSGridHeader   | PSNavigator | PSTree         |
| PSGridHeaderVP | PSPanel     | PSTreeHeader   |
| PSGridVP       | PSPanelVP   | PSTreeHeaderVP |
| PSMenu         | PSSpin      | PSTreeVP       |

## PSPanel

### User Action Command

»»SQA»

### Description

Performs an action on a PeopleTools panel. The panel can be encountered at application runtime or while you're editing the panel in the PeopleTools Panel Designer.

### Syntax

**PSPanel** *action%*, *recMethod\$*, *parameters\$*

| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i> | One of these actions: <ul style="list-style-type: none"><li>▶ <i>MouseClicked</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> must contain <i>Coords=x,y</i>.</li><li>▶ <i>MouseDown</i>. The dragging of the mouse while mouse buttons and/or shifting keys (Ctrl, Alt, Shift) are pressed. When <i>action%</i> contains a mouse-drag value, <i>parameters\$</i> must contain <i>Coords=x1,y1,x2,y2</i>.</li></ul> See Appendix E for a list of mouse click and drag values. |

▶ ▶ ▶

▶ ▶ ▶

| Syntax Element     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                 |                |             |                |                |              |            |              |           |           |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|----------------|-------------|----------------|----------------|--------------|------------|--------------|-----------|-----------|
|                    | <p>▶ <i>ScrollAction</i>. One of these scroll actions:</p> <table border="0" data-bbox="808 485 1274 625"> <tr> <td>ScrollPageRight</td> <td>ScrollPageDown</td> </tr> <tr> <td>ScrollRight</td> <td>ScrollLineDown</td> </tr> <tr> <td>ScrollPageLeft</td> <td>ScrollPageUp</td> </tr> <tr> <td>ScrollLeft</td> <td>ScrollLineUp</td> </tr> <tr> <td>HScrollTo</td> <td>VScrollTo</td> </tr> </table> <p>HScrollTo and VScrollTo take the required parameter <code>Position=%</code>.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | ScrollPageRight | ScrollPageDown | ScrollRight | ScrollLineDown | ScrollPageLeft | ScrollPageUp | ScrollLeft | ScrollLineUp | HScrollTo | VScrollTo |
| ScrollPageRight    | ScrollPageDown                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                 |                |             |                |                |              |            |              |           |           |
| ScrollRight        | ScrollLineDown                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                 |                |             |                |                |              |            |              |           |           |
| ScrollPageLeft     | ScrollPageUp                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                 |                |             |                |                |              |            |              |           |           |
| ScrollLeft         | ScrollLineUp                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                 |                |             |                |                |              |            |              |           |           |
| HScrollTo          | VScrollTo                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |                 |                |             |                |                |              |            |              |           |           |
| <i>recMethod\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>FieldID=%</code>. A unique ID that identifies a particular <i>field</i> on a panel. Used only after one of these parent values: <code>ID=%</code>, <code>Name=\$</code>, <code>ObjectIndex=%</code>, <code>Text=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>;\;</code>). See <i>Comments</i> for more information.</li> <li>▶ <code>FieldIndex=%</code>. A numeric value used to distinguish between multiple panel <i>fields</i> with the same name. Used only after one of these parent values: <code>ID=%</code>, <code>Name=\$</code>, <code>ObjectIndex=%</code>, <code>Text=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>;\;</code>). See <i>Comments</i> for more information.</li> <li>▶ <code>FieldLabel=\$</code>. The displayed name of a particular <i>field</i> on a panel. Used only after one of these parent values: <code>ID=%</code>, <code>Name=\$</code>, <code>ObjectIndex=%</code>, <code>Text=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>;\;</code>). See <i>Comments</i> for more information.</li> <li>▶ <code>ID=%</code>. The object's internal Windows ID.</li> <li>▶ <code>Name=\$</code>. A name that a developer assigns to an object to uniquely identify the object in the development environment. Name can specify the name of the panel or an object on the panel.</li> <li>▶ When Name specifies an object on the panel, it is used only after one of these parent values: <code>ID=%</code>, <code>Name=\$</code>, <code>ObjectIndex=%</code>, <code>Text=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>;\;</code>).</li> <li>▶ <code>ObjectIndex=%</code>. The number of the object among all objects of the same type in the same window.</li> </ul> |                 |                |             |                |                |              |            |              |           |           |

▶ ▶ ▶

## PSPanel

▶ ▶ ▶

| Syntax Element      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                     | <ul style="list-style-type: none"> <li>▶ <code>State=\$</code>. An optional qualifier for any other recognition method. There are two possible values for this setting: <code>Enabled</code> and <code>Disabled</code>. The default state is the state of the current context window (as set in the most recent <code>Window SetContext</code> command), or <code>Enabled</code> if the state has not been otherwise declared.</li> <li>▶ <code>Text=\$</code>. The text displayed on the object.</li> <li>▶ <code>VisualText=\$</code>. An optional setting used to identify an object by its visible text. It is for user clarification only and does not affect object recognition.</li> </ul>                                                                                                                       |
| <i>parameters\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>Coords=x,y</code>. If <i>action%</i> is a mouse click, specifies the <i>x,y</i> coordinates of the click, relative to the top left of the object or the item.</li> <li>▶ <code>Coords=x1,y1,x2,y2</code>. If <i>action%</i> is a mouse drag, specifies the coordinates, where <i>x1,y1</i> are the starting coordinates of the drag, and <i>x2,y2</i> are the ending coordinates. The coordinates are relative to the top left of the object or the item.</li> <li>▶ <code>Position=%</code>. If <i>action%</i> is <code>VScrollTo</code> or <code>HScrollTo</code>, specifies the scroll bar value of the new scrolled-to position in the scroll box. Every scroll bar has an internal range, and this value is specific to that range.</li> </ul> |

## Comments

Robot can recognize all objects (fields) within a PeopleTools panel and can test each field object according to its type. Robot recognizes the following PeopleSoft field objects (names in parentheses are the associated SQA object names):

|                               |                              |
|-------------------------------|------------------------------|
| Check Box                     | Push Button                  |
| Drop Down List Box (ComboBox) | Radio Button                 |
| Edit Box                      | Scroll Bar                   |
| Frame                         | Secondary Panel              |
| Group Box                     | Static Image (GenericObject) |
| Image (GenericObject)         | SubPanel                     |
| Long Edit Box (EditBox)       | Text (Label)                 |

Note that SubPanels and Secondary panels are only available when you're editing a panel in the Panel Designer. These objects might not exist at application runtime.



To uniquely identify fields that appear as objects on panels, Robot uses one of these identifiers:

Combined record/field name – In some cases, a field that appears on a panel is associated with a field in a PeopleTools database. Robot constructs a unique *panel object* name for these fields by combining the database record name that the field appears in plus the field name. The record name and field name are separated by a period character ( . ). For example, the following recognition method uniquely identifies the ABSENCE\_TYPE field within the ABSENCE\_HIST record as a panel object (an edit box):

```
EditBox Click, "Name=ABSENCE_HIST.ABSENCE_TYPE", "Coords=113,11"
```

Record/field syntax has these additional features:

- When *multiple occurrences* of the same field appear within a panel (for example, when the OccursCount for an associated Scroll Bar is greater than 1), Robot distinguishes each field through a zero-based index value. The index value appears in parentheses after the record/field name. For example, suppose an Edit Box field is associated with a scroll bar with an OccursCount of 3. In the database, the Edit Box field, named EMPLOYEEES, is in a record named ABSENCE\_HIST. The following recognition method identifies the second occurrence of the Edit Box on the panel:

```
EditBox Click, "Name=ABSENCE_HIST.EMPLOYEEES(1)", "Coords=101,9"
```

- If a field on a panel is a *Related Display* (that is, its value is derived from the value of another field), Robot uses the record/field names of the source field as part of the destination field's name. The record/field names are separated by a pointer ( -> ). For example, if the value of field PANEL2.FIELD2 is derived from the value of PANEL1.FIELD1, Robot identifies FIELD 2 as follows:

```
PANEL1.FIELD1->PANEL2.FIELD2
```

FieldLabel – If multiple fields on a panel have the same record/field name and occurrence index (such as groups of radio buttons), Robot identifies the field through both its panel object name and its field *label* (or possibly just its field label if there is no associated record/field). For example:

```
RadioButton Click, "FieldLabel=Female;Name=Personal_Data.Sex"
```

If Robot can't recognize a field by a record/field name or a field label, it uses FieldIndex or FieldID identifiers.

FieldIndex – This number specifies a particular field within a field *type*. For example, the second GroupBox on a panel might be recognized as FieldIndex=2. Field index numbers begin with 1.

FieldID – This number is a unique zero-based identifier assigned to each field object in a panel. For example;

```
ScrollBar Click, "FieldID=9", "Coords=9,68"
```

PSPanelVP

**Example** This example clicks a panel with the internal object name ABSENCE\_HISTORY.

```
PSPanel Click, "Name=ABSENCE_HISTORY", "Coords=10,5"
```

**See Also**

|                |               |                |
|----------------|---------------|----------------|
| PSGrid         | PSMenuVP      | PSSpinVP       |
| PSGridHeader   | PSNavigator   | PSTree         |
| PSGridHeaderVP | PSNavigatorVP | PSTreeHeader   |
| PSGridVP       | PSPanelVP     | PSTreeHeaderVP |
| PSMenu         | PSSpin        | PSTreeVP       |

## PSPanelVP

Verification Point Command

»»SQ»

**Description** Establishes a verification point for a PeopleTools panel. The panel can be encountered at application runtime or while you're editing the panel in the PeopleTools Panel Designer.

**Syntax** *Result* = **PSPanelVP** (*action%*, *recMethod\$*, *parameters\$*)

| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i> | <p>The type of verification to perform. Valid values:</p> <ul style="list-style-type: none"><li>▶ <b>CompareData</b>. Captures the data of the object and compares it to a recorded baseline. <i>parameters\$ VP</i> and <b>Type</b> are required; <b>ExpectedResult</b> and <b>Wait</b> are optional.</li><li>▶ <b>CompareNumeric</b>. Captures the numeric value of the text of the object and compares it to the value of <i>parameters\$ Value or Range</i>. <i>parameters\$ VP</i> and either <b>Value or Range</b> are required; <b>ExpectedResult</b> and <b>Wait</b> are optional.</li><li>▶ <b>CompareProperties</b>. Captures object properties information for the object and compares it to a recorded baseline. <i>parameters\$ VP</i> is required; <b>ExpectedResult</b> and <b>Wait</b> are optional.</li><li>▶ <b>CompareText</b>. Captures the text of the object and compares it to a recorded baseline. <i>parameters\$ VP</i> and <b>Type</b> are required; <b>ExpectedResult</b> and <b>Wait</b> are optional.</li></ul> |

▶ ▶ ▶

▶ ▶ ▶

| Syntax Element       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>recMethod</i> \$  | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <i>FieldID</i>=%. A unique ID that identifies a particular <i>field</i> on a panel. Used only after one of these parent values: <i>ID</i>=%, <i>Name</i>=\$, <i>ObjectIndex</i>=%, <i>Text</i>=\$. Parent/child values are separated by a backslash and semicolons (; \ ;). See <i>Comments</i> for more information.</li> <li>▶ <i>FieldIndex</i>=%. A numeric value used to distinguish between multiple panel <i>fields</i> with the same name. Used only after one of these parent values: <i>ID</i>=%, <i>Name</i>=\$, <i>ObjectIndex</i>=%, <i>Text</i>=\$. Parent/child values are separated by a backslash and semicolons (; \ ;). See <i>Comments</i> for more information.</li> <li>▶ <i>FieldLabel</i>=\$. The displayed name of a particular <i>field</i> on a panel. Used only after one of these parent values: <i>ID</i>=%, <i>Name</i>=\$, <i>ObjectIndex</i>=%, <i>Text</i>=\$. Parent/child values are separated by a backslash and semicolons (; \ ;). See <i>Comments</i> for more information.</li> <li>▶ <i>ID</i>=%. The object's internal Windows ID.</li> <li>▶ <i>Name</i>=\$. A name that a developer assigns to an object to uniquely identify the object in the development environment. <i>Name</i> can specify the name of the panel or an object on the panel.</li> <li>▶ When <i>Name</i> specifies an object on the panel, it is used only after one of these parent values: <i>ID</i>=%, <i>Name</i>=\$, <i>ObjectIndex</i>=%, <i>Text</i>=\$. Parent/child values are separated by a backslash and semicolons (; \ ;).</li> <li>▶ <i>ObjectIndex</i>=%. The number of the object among all objects of the same type in the same window.</li> <li>▶ <i>Text</i>=\$. The text displayed on the object.</li> </ul> |
| <i>parameters</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <i>ExpectedResult</i>=%. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> <li>– <i>PASS</i>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail.</li> <li>– <i>FAIL</i>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail.</li> </ul> </li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |

▶ ▶ ▶



| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                | <ul style="list-style-type: none"> <li>▶ <code>Range=&amp; , &amp;</code>. Used with the action <code>CompareNumeric</code> when a numeric range comparison is being performed, as in <code>Range=2 , 12</code> (test for numbers in this range). The values are inclusive.</li> <li>▶ <code>Type=\$</code>. Specifies the verification method to use for <code>CompareText</code> actions. The possible values are: <code>CaseSensitive</code>, <code>CaseInsensitive</code>, <code>FindSubStr</code>, <code>FindSubStrI</code> (case insensitive), and <code>UserDefined</code>. See <i>Comments</i> for more information. If <code>UserDefined</code> is specified, two additional parameters are required: <ul style="list-style-type: none"> <li>- <code>DLL=\$</code>. The full path and file name of the library that contains the function</li> <li>- <code>Function=\$</code>. The name of the custom function to use in comparing the text</li> </ul> </li> <li>▶ <code>Value=&amp;</code>. Used with the action <code>CompareNumeric</code> when a numeric equivalence comparison is being performed, as in <code>Value=25</code> (test against the value 25).</li> <li>▶ <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points.</li> <li>▶ <code>Wait=% , %</code>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10 , 40</code> (retry the test every 10 seconds, but time out the test after 40 seconds).</li> </ul> |

**Comments** This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

With the `Type=$` parameter, `CaseSensitive` and `CaseInsensitive` require a full match between the current baseline text and the text captured during playback. With `FindSubStr` and `FindSubStrI`, the current baseline can be a substring of the text captured during playback. The substring can appear anywhere in the playback text. To modify the current baseline text, double-click the verification point name in the Robot Asset pane (to the left of the script).

Robot can recognize all objects (fields) within a PeopleTools panel and can test each field object according to its type. Robot recognizes the following PeopleSoft field objects (names in parentheses are the associated SQA object names):

|                               |                              |
|-------------------------------|------------------------------|
| Check Box                     | Push Button                  |
| Drop Down List Box (ComboBox) | Radio Button                 |
| Edit Box                      | Scroll Bar                   |
| Frame                         | Secondary Panel              |
| Group Box                     | Static Image (GenericObject) |
| Image (GenericObject)         | SubPanel                     |
| Long Edit Box (EditBox)       | Text (Label)                 |

Note that SubPanels and Secondary panels are only available when you're editing a panel in the Panel Designer. These objects might not exist at application runtime.

To uniquely identify fields that appear as objects on panels, Robot uses one of these identifiers:

Combined record/field name – In some cases, a field that appears on a panel is associated with a field in a PeopleTools database. Robot constructs a unique *panel object* name for these fields by combining the database record name that the field appears in plus the field name. The record name and field name are separated by a period character (.). For example, the following recognition method uniquely identifies the ABSENCE\_TYPE field within the ABSENCE\_HIST record as a panel object:

```
Result = EditBoxVP, (CompareProperties,
 "Name=ABSENCE_HIST.ABSENCE_TYPE", "VP=EMPABS")
```

Record/field syntax has these additional features:

- When *multiple occurrences* of the same field appear within a panel (for example, when the OccursCount for an associated Scroll Bar is greater than 1), Robot distinguishes each field through a zero-based index value. The index value appears in parentheses after the record/field name. For example, suppose an Edit Box field is associated with a scroll bar with an OccursCount of 3. In the database, the Edit Box field, named EMPLOYEEES, is in a record named ABSENCE\_HIST. The following recognition method identifies the second occurrence of the Edit Box on the panel:

```
Result = EditBoxVP (CompareProperties,
 "Name=ABSENCE_HIST.EMPLOYEEES(1)", "VP=EMPNAME")
```

- If a field on a panel is a *Related Display* (that is, its value is derived from the value of another field), Robot uses the record/field names of the source field as part of the destination field's name. The record/field names are separated by a pointer (->). For example, if the value of field PANEL2.FIELD2 is derived from the value of PANEL1.FIELD1, Robot identifies FIELD 2 as follows:

```
PANEL1.FIELD1->PANEL2.FIELD2
```

FieldLabel – If multiple fields on a panel have the same record/field name and occurrence index (such as groups of radio buttons), Robot identifies the field through both its panel object name and its field *label* (or possibly just its field label if there is no associated record/field). For example:

```
Result = RadioButonVP (CompareProperties,
 "FieldLabel=Female;Name=Personal_Data.Sex", "VP=EMPABSNC")
```

If Robot can't recognize a field by a record/field name or a field label, it uses FieldIndex or FieldID identifiers.

## PSSpin

**FieldIndex** – This number specifies a particular field within a field *type*. For example, the second GroupBox on a panel might be recognized as FieldIndex=2. Field index numbers begin with 1.

**FieldID** – This number is a unique zero-based identifier assigned to each field object in a panel. For example:

```
Result=ScrollBarVP(CompareProperties,"FieldID=4","VP=SCRLPROP")
```

### Example

This example establishes an object properties verification point for the PeopleTools panel with the internal object name AE\_REQUEST.

```
Result = PPanelVP (CompareProperties, "Name=AE_REQUEST", "VP=QRY4")
```

### See Also

|                |               |                |
|----------------|---------------|----------------|
| PSGrid         | PSMenuVP      | PSSpinVP       |
| PSGridHeader   | PSNavigator   | PSTree         |
| PSGridHeaderVP | PSNavigatorVP | PSTreeHeader   |
| PSGridVP       | PSPanel       | PSTreeHeaderVP |
| PSMenu         | PSSpin        | PSTreeVP       |

## PSSpin

User Action Command

»»SQA»

**Description** Performs an action on a PeopleTools spin control.

**Syntax** **PSSpin** *action%*, *recMethod\$*, *parameters\$*

| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i> | One of these actions: <ul style="list-style-type: none"><li>▶ <i>MouseClicked</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> must contain <i>Coords=x,y</i>.</li><li>▶ <i>MouseDown</i>. The dragging of the mouse while mouse buttons and/or shifting keys (Ctrl, Alt, Shift) are pressed. When <i>action%</i> contains a mouse-drag value, <i>parameters\$</i> must contain <i>Coords=x1,y1,x2,y2</i>. See Appendix E for a list of mouse click and drag values.</li></ul> |

▶ ▶ ▶

▶ ▶ ▶

| Syntax Element      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                 |                |             |                |                |              |            |              |           |           |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|----------------|-------------|----------------|----------------|--------------|------------|--------------|-----------|-----------|
|                     | <p>▶ <i>ScrollAction</i>. One of these scroll actions:</p> <table border="0"> <tr> <td>ScrollPageRight</td> <td>ScrollPageDown</td> </tr> <tr> <td>ScrollRight</td> <td>ScrollLineDown</td> </tr> <tr> <td>ScrollPageLeft</td> <td>ScrollPageUp</td> </tr> <tr> <td>ScrollLeft</td> <td>ScrollLineUp</td> </tr> <tr> <td>HScrollTo</td> <td>VScrollTo</td> </tr> </table> <p>HScrollTo and VScrollTo take the required parameter <i>Position=%</i>.</p> <p>If Robot cannot interpret the action being applied to a scroll bar, which happens with certain custom standalone scroll bars, it records the action as a click or drag.</p>                                                                                                                                                                                                                                                                                                               | ScrollPageRight | ScrollPageDown | ScrollRight | ScrollLineDown | ScrollPageLeft | ScrollPageUp | ScrollLeft | ScrollLineUp | HScrollTo | VScrollTo |
| ScrollPageRight     | ScrollPageDown                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                 |                |             |                |                |              |            |              |           |           |
| ScrollRight         | ScrollLineDown                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                 |                |             |                |                |              |            |              |           |           |
| ScrollPageLeft      | ScrollPageUp                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                 |                |             |                |                |              |            |              |           |           |
| ScrollLeft          | ScrollLineUp                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                 |                |             |                |                |              |            |              |           |           |
| HScrollTo           | VScrollTo                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                 |                |             |                |                |              |            |              |           |           |
| <i>recMethod\$</i>  | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <i>ID=%</i>. The object's internal Windows ID.</li> <li>▶ <i>Label=\$</i>. The text of the label object that immediately precedes the PeopleSoft spin control in the internal order (Z order) of windows.</li> <li>▶ <i>ObjectIndex=%</i>. The number of the object among all objects of the same type in the same window.</li> <li>▶ <i>State=\$</i>. An optional qualifier for any other recognition method. There are two possible values for this setting: <i>Enabled</i> and <i>Disabled</i>. The default state is the state of the current context window (as set in the most recent <i>Window SetContext</i> command), or <i>Enabled</i> if the state has not been otherwise declared.</li> <li>▶ <i>VisualText=\$</i>. An optional setting used to identify an object by its prior label. It is for user clarification only and does not affect object recognition.</li> </ul> |                 |                |             |                |                |              |            |              |           |           |
| <i>parameters\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <i>Coords=x,y</i>. If <i>action%</i> is a mouse click, specifies the <i>x,y</i> coordinates of the click, relative to the top left of the object or the item.</li> <li>▶ <i>Coords=x1,y1,x2,y2</i>. If <i>action%</i> is a mouse drag, specifies the coordinates, where <i>x1,y1</i> are the starting coordinates of the drag, and <i>x2,y2</i> are the ending coordinates. The coordinates are relative to the top left of the object or the item.</li> <li>▶ <i>Position=%</i>. If <i>action%</i> is <i>VScrollTo</i> or <i>HScrollTo</i>, specifies the scroll bar value of the new scrolled-to position in the scroll box. Every scroll bar has an internal range, and this value is specific to that range.</li> </ul>                                                                                                                                                            |                 |                |             |                |                |              |            |              |           |           |

PSSpinVP

Comments None.

Example None.

See Also

|                |               |                |
|----------------|---------------|----------------|
| PSGrid         | PSMenuVP      | PSSpinVP       |
| PSGridHeader   | PSNavigator   | PSTree         |
| PSGridHeaderVP | PSNavigatorVP | PSTreeHeader   |
| PSGridVP       | PSPanel       | PSTreeHeaderVP |
| PSMenu         | PSPanelVP     | PSTreeVP       |

## PSSpinVP

Verification Point Command

»»SQA»

Description Establishes a verification point for a PeopleTools spin control.

Syntax *Result* = **PSSpinVP** (*action%*, *recMethod\$*, *parameters\$*)

| Syntax Element     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i>     | <p>The type of verification to perform. Valid values:</p> <ul style="list-style-type: none"><li>▶ <b>CompareNumeric</b>. Captures the numeric value of the text of the object and compares it to the value of <i>parameters\$</i> Value or Range. <i>parameters\$</i> VP and either Value or Range are required; ExpectedResult and Wait are optional.</li><li>▶ <b>CompareProperties</b>. Captures object properties information for the object and compares it to a recorded baseline. <i>parameters\$</i> VP is required; ExpectedResult and Wait are optional.</li><li>▶ <b>CompareText</b>. Captures the text of the object and compares it to a recorded baseline. <i>parameters\$</i> VP and Type are required; ExpectedResult and Wait are optional.</li></ul> |
| <i>recMethod\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"><li>▶ <b>ID=%</b>. The object's internal Windows ID.</li><li>▶ <b>Label=\$</b>. The text of the label object that immediately precedes the PeopleSoft spin control in the internal order (Z order) of windows.</li><li>▶ <b>Name=\$</b>. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be Command1.</li><li>▶ <b>ObjectIndex=%</b>. The number of the object among all objects of the same type in the same window.</li></ul> <p style="text-align: right;">▶ ▶ ▶</p>                                                                                                                    |



▶ ▶ ▶

| Syntax Element            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|---------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>parameters\$</code> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>ExpectedResult=%</code>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> <li>– <code>PASS</code>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail.</li> <li>– <code>FAIL</code>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail.</li> </ul> </li> <li>▶ <code>Range=&amp; , &amp;</code>. Used with the action <code>CompareNumeric</code> when a numeric range comparison is being performed, as in <code>Range=2 , 12</code> (test for numbers in this range). The values are inclusive.</li> <li>▶ <code>Type=\$</code>. Specifies the verification method to use for <code>CompareText</code> actions. The possible values are: <code>CaseSensitive</code>, <code>CaseInsensitive</code>, <code>FindSubStr</code>, <code>FindSubStrI</code> (case insensitive), and <code>UserDefined</code>. See <i>Comments</i> for more information. If <code>UserDefined</code> is specified, two additional parameters are required: <ul style="list-style-type: none"> <li>– <code>DLL=\$</code>. The full path and file name of the library that contains the function</li> <li>– <code>Function=\$</code>. The name of the custom function to use in comparing the text</li> </ul> </li> <li>▶ <code>Value=&amp;</code>. Used with the action <code>CompareNumeric</code> when a numeric equivalence comparison is being performed, as in <code>Value=25</code> (test against the value 25).</li> <li>▶ <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points.</li> <li>▶ <code>Wait=% , %</code>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10 , 40</code> (retry the test every 10 seconds, but time out the test after 40 seconds).</li> </ul> |

**Comments** This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

## PSTree

With the `Type=$` parameter, `CaseSensitive` and `CaseInsensitive` require a full match between the current baseline text and the text captured during playback. With `FindSubStr` and `FindSubStrI`, the current baseline can be a substring of the text captured during playback. The substring can appear anywhere in the playback text. To modify the current baseline text, double-click the verification point name in the Robot Asset pane (to the left of the script).

Example      None.

See Also

|                             |                            |                             |
|-----------------------------|----------------------------|-----------------------------|
| <code>PSGrid</code>         | <code>PSMenuVP</code>      | <code>PSSpin</code>         |
| <code>PSGridHeader</code>   | <code>PSNavigator</code>   | <code>PSTree</code>         |
| <code>PSGridHeaderVP</code> | <code>PSNavigatorVP</code> | <code>PSTreeHeader</code>   |
| <code>PSGridVP</code>       | <code>PSPanel</code>       | <code>PSTreeHeaderVP</code> |
| <code>PSMenu</code>         | <code>PSPanelVP</code>     | <code>PSTreeVP</code>       |

## PSTree

User Action Command

»»SQA»

Description      Performs an action on a PeopleTools tree object.

Syntax            **PSTree** *action%*, *recMethod\$*, *parameters\$*

| Syntax Element               | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                              |                             |                          |                             |                             |                           |                         |                           |                        |                        |
|------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------|-----------------------------|--------------------------|-----------------------------|-----------------------------|---------------------------|-------------------------|---------------------------|------------------------|------------------------|
| <i>action%</i>               | <p>One of these actions:</p> <ul style="list-style-type: none"><li>▶ <i>MouseClicked</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> must contain <code>Coords=x,y</code>.</li><li>▶ <i>MouseDown</i>. The dragging of the mouse while mouse buttons and/or shifting keys (Ctrl, Alt, Shift) are pressed. When <i>action%</i> contains a mouse-drag value, <i>parameters\$</i> must contain <code>Coords=x1,y1,x2,y2</code>. See Appendix E for a list of mouse click and drag values.</li><li>▶ <i>ScrollAction</i>. One of these scroll actions:<table><tbody><tr><td><code>ScrollPageRight</code></td><td><code>ScrollPageDown</code></td></tr><tr><td><code>ScrollRight</code></td><td><code>ScrollLineDown</code></td></tr><tr><td><code>ScrollPageLeft</code></td><td><code>ScrollPageUp</code></td></tr><tr><td><code>ScrollLeft</code></td><td><code>ScrollLineUp</code></td></tr><tr><td><code>HScrollTo</code></td><td><code>VScrollTo</code></td></tr></tbody></table><p><code>HScrollTo</code> and <code>VScrollTo</code> take the required parameter <code>Position=%</code>.</p></li></ul> | <code>ScrollPageRight</code> | <code>ScrollPageDown</code> | <code>ScrollRight</code> | <code>ScrollLineDown</code> | <code>ScrollPageLeft</code> | <code>ScrollPageUp</code> | <code>ScrollLeft</code> | <code>ScrollLineUp</code> | <code>HScrollTo</code> | <code>VScrollTo</code> |
| <code>ScrollPageRight</code> | <code>ScrollPageDown</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                              |                             |                          |                             |                             |                           |                         |                           |                        |                        |
| <code>ScrollRight</code>     | <code>ScrollLineDown</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                              |                             |                          |                             |                             |                           |                         |                           |                        |                        |
| <code>ScrollPageLeft</code>  | <code>ScrollPageUp</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                              |                             |                          |                             |                             |                           |                         |                           |                        |                        |
| <code>ScrollLeft</code>      | <code>ScrollLineUp</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                              |                             |                          |                             |                             |                           |                         |                           |                        |                        |
| <code>HScrollTo</code>       | <code>VScrollTo</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                              |                             |                          |                             |                             |                           |                         |                           |                        |                        |

▶ ▶ ▶

▶ ▶ ▶

| Syntax Element       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>recMethod</i> \$  | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <i>ID</i>=%. The object's internal Windows ID.</li> <li>▶ <i>ItemIndex</i>=%. The index of the tree view item acted upon. Used only after one of these parent values: <i>ID</i>=%, <i>ObjectIndex</i>=%, <i>Text</i>=\$. Parent/child values are separated by a backslash and semicolons (<i>i</i>\;<i>i</i>).</li> <li>▶ <i>ItemText</i>=\$. The text of the tree view item acted upon. Used only after one of these parent values: <i>ID</i>=%, <i>ObjectIndex</i>=%, <i>Text</i>=\$. Parent/child values are separated by a backslash and semicolons (<i>i</i>\;<i>i</i>).</li> <li>▶ <i>ObjectIndex</i>=%. The number of the object among all objects of the same type in the same window.</li> <li>▶ <i>State</i>=\$. An optional qualifier for any other recognition method. There are two possible values for this setting: <i>Enabled</i> and <i>Disabled</i>. The default state is the state of the current context window (as set in the most recent <i>Window SetContext</i> command), or <i>Enabled</i> if the state has not been otherwise declared.</li> <li>▶ <i>Text</i>=\$. The text displayed on the object.</li> <li>▶ <i>VisualText</i>=\$. An optional setting used to identify an object by its visible text. It is for user clarification only and does not affect object recognition.</li> </ul> |
| <i>parameters</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <i>Coords</i>=<i>x,y</i>. If <i>action</i>% is a mouse click, specifies the <i>x,y</i> coordinates of the click, relative to the top left of the object or the item.</li> <li>▶ <i>Coords</i>=<i>x1,y1,x2,y2</i>. If <i>action</i>% is a mouse drag, specifies the coordinates, where <i>x1,y1</i> are the starting coordinates of the drag, and <i>x2,y2</i> are the ending coordinates. The coordinates are relative to the top left of the object or the item.</li> <li>▶ <i>Position</i>=%. If <i>action</i>% is <i>VScrollTo</i> or <i>HScrollTo</i>, specifies the scroll bar value of the new scrolled-to position in the scroll box. Every scroll bar has an internal range, and this value is specific to that range.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

## PSTreeHeader

**Comments** The tree object can appear in the PeopleTools Tree Manager environment.

When you act on a particular item in a tree object, Robot uses the text of the item (plus the text of any parent items) to identify it. In the following *recMethod\$* value, the tree item labeled REPORT\_VIEWS is a child of the tree item labeled HR\_ACCESS\_GROUP.

```
"ObjectIndex=1;\;ItemText=HR_ACCESS_GROUP->REPORT_VIEWS"
```

Note the two different parent/child separators — the *backslash* (\) separates the window object and its child object. The *pointer* (->) separates the parent text item from its child text item in the tree hierarchy.

**Example** This example clicks an item in a PeopleTools tree object. The clicked item is identified by the text 10100.

```
PSTree Click, "ObjectIndex=1;\;ItemText=00001->10100",
"Coords=19,12"
```

## See Also

|                |               |                |
|----------------|---------------|----------------|
| PSGrid         | PSMenuVP      | PSSpin         |
| PSGridHeader   | PSNavigator   | PSSpinVP       |
| PSGridHeaderVP | PSNavigatorVP | PSTreeHeader   |
| PSGridVP       | PSPanel       | PSTreeHeaderVP |
| PSMenu         | PSPanelVP     | PSTreeVP       |

## PSTreeHeader

### User Action Command



**Description** Performs an action on a column header in a PeopleTools tree object.

**Syntax** **PSTreeHeader** *action%*, *recMethod\$*, *parameters\$*

| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i> | One of these mouse actions: <ul style="list-style-type: none"><li>▶ <i>MouseClicked</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> must contain <i>Coords=x,y</i>.</li><li>▶ <i>MouseDown</i>. The dragging of the mouse while mouse buttons and/or shifting keys (Ctrl, Alt, Shift) are pressed. When <i>action%</i> contains a mouse-drag value, <i>parameters\$</i> must contain <i>Coords=x1,y1,x2,y2</i>.</li></ul> See Appendix E for a list of mouse click and drag values. |





| Syntax Element       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>recMethod</i> \$  | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ ID=%. The object's internal Windows ID.</li> <li>▶ ObjectIndex=%. The number of the object among all objects of the same type in the same window.</li> <li>▶ State=\$. An optional qualifier for any other recognition method. There are two possible values for this setting: <code>Enabled</code> and <code>Disabled</code>. The default state is the state of the current context window (as set in the most recent <code>Window SetContext</code> command), or <code>Enabled</code> if the state has not been otherwise declared.</li> <li>▶ Text=\$. The text displayed on the object.</li> <li>▶ VisualText=\$. An optional setting used to identify an object by its visible text. It is for user clarification only and does not affect object recognition.</li> </ul> |
| <i>parameters</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ Coords=<i>x,y</i>. If <i>action%</i> is a mouse click, specifies the <i>x,y</i> coordinates of the click, relative to the top left of the object or the item.</li> <li>▶ Coords=<i>x1,y1,x2,y2</i>. If <i>action%</i> is a mouse drag, specifies the coordinates, where <i>x1,y1</i> are the starting coordinates of the drag, and <i>x2,y2</i> are the ending coordinates. The coordinates are relative to the top left of the object or the item.</li> </ul>                                                                                                                                                                                                                                                                                                                 |

**Comments** The tree object can appear in the PeopleTools Tree Manager environment.

**Example** In this example, the user double-clicks a tree header. The tree object is identified as object 1 in the current context window.

```
PSTreeHeader DblClick, "ObjectIndex=1", "Coords=232,9"
```

**See Also**

|                |               |                |
|----------------|---------------|----------------|
| PSGrid         | PSMenuVP      | PSSpin         |
| PSGridHeader   | PSNavigator   | PSSpinVP       |
| PSGridHeaderVP | PSNavigatorVP | PSTree         |
| PSGridVP       | PSPanel       | PSTreeHeaderVP |
| PSMenu         | PSPanelVP     | PSTreeVP       |

## PSTreeHeaderVP

Verification Point Command

**Description** Establishes a verification point for a column header in a PeopleTools tree object.

**Syntax** `Result = PSTreeHeaderVP (action%, recMethod$, parameters$)`

| Syntax Element     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i>     | <p>The type of verification to perform. Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>CompareData</code>. Captures the data of the object and compares it to a recorded baseline. <i>parameters\$ VP</i> and <code>Type</code> are required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li> <li>▶ <code>CompareNumeric</code>. Captures the numeric value of the text of the object and compares it to the value of <i>parameters\$ Value or Range</i>. <i>parameters\$ VP</i> and either <code>Value or Range</code> are required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li> <li>▶ <code>CompareProperties</code>. Captures object properties information for the object and compares it to a recorded baseline. <i>parameters\$ VP</i> is required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li> <li>▶ <code>CompareText</code>. Captures the text of the object and compares it to a recorded baseline. <i>parameters\$ VP</i> and <code>Type</code> are required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li> </ul> |
| <i>recMethod\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>ID=%</code>. The object's internal Windows ID.</li> <li>▶ <code>Name=\$</code>. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be <code>Command1</code>.</li> <li>▶ <code>ObjectIndex=%</code>. The number of the object among all objects of the same type in the same window.</li> <li>▶ <code>Text=\$</code>. The text displayed on the object.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |

▶ ▶ ▶

▶ ▶ ▶

| Syntax Element            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|---------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>parameters\$</code> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>ExpectedResult=%</code>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> <li>– <code>PASS</code>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail.</li> <li>– <code>FAIL</code>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail.</li> </ul> </li> <li>▶ <code>Range=&amp;, &amp;</code>. Used with the action <code>CompareNumeric</code> when a numeric range comparison is being performed, as in <code>Range=2, 12</code> (test for numbers in this range). The values are inclusive.</li> <li>▶ <code>Type=\$</code>. Specifies the verification method to use for <code>CompareText</code> actions. The possible values are: <code>CaseSensitive</code>, <code>CaseInsensitive</code>, <code>FindSubStr</code>, <code>FindSubStrI</code> (case insensitive), and <code>UserDefined</code>. See <i>Comments</i> for more information. If <code>UserDefined</code> is specified, two additional parameters are required: <ul style="list-style-type: none"> <li>– <code>DLL=\$</code>. The full path and file name of the library that contains the function</li> <li>– <code>Function=\$</code>. The name of the custom function to use in comparing the text</li> </ul> </li> <li>▶ <code>Value=&amp;</code>. Used with the action <code>CompareNumeric</code> when a numeric equivalence comparison is being performed, as in <code>Value=25</code> (test against the value 25).</li> <li>▶ <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points.</li> <li>▶ <code>Wait=%, %</code>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10, 40</code> (retry the test every 10 seconds, but time out the test after 40 seconds).</li> </ul> |

**Comments** This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

With the `Type=$` parameter, `CaseSensitive` and `CaseInsensitive` require a full match between the current baseline text and the text captured during playback. With `FindSubStr` and `FindSubStrI`, the current baseline can be a

## PSTreeVP

substring of the text captured during playback. The substring can appear anywhere in the playback text. To modify the current baseline text, double-click the verification point name in the Robot Asset pane (to the left of the script).

The tree object can appear in the PeopleTools Tree Manager environment.

**Example** This example establishes an object properties verification point for a tree header. The tree object is identified as object 1 in the current context window.

```
Result = PSTreeHeaderVP (CompareProperties, "ObjectIndex=1",
"VP=COLTST")
```

## See Also

|                |               |              |
|----------------|---------------|--------------|
| PSGrid         | PSMenuVP      | PSSpin       |
| PSGridHeader   | PSNavigator   | PSSpinVP     |
| PSGridHeaderVP | PSNavigatorVP | PSTree       |
| PSGridVP       | PSPanel       | PSTreeHeader |
| PSMenu         | PSPanelVP     | PSTreeVP     |

## PSTreeVP

### Verification Point Command



**Description** Establishes a verification point for a PeopleTools tree object.

**Syntax** `Result = PSTreeVP (action%, recMethod$, parameters$)`

| Syntax Element       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>action%</code> | <p>The type of verification to perform. Valid values:</p> <ul style="list-style-type: none"><li>▶ <code>CompareData</code>. Captures the data of the object and compares it to a recorded baseline. <code>parameters\$ VP</code> and <code>Type</code> are required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li><li>▶ <code>CompareNumeric</code>. Captures the numeric value of the text of the object and compares it to the value of <code>parameters\$ Value or Range</code>. <code>parameters\$ VP</code> and either <code>Value or Range</code> are required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li><li>▶ <code>CompareProperties</code>. Captures object properties information for the object and compares it to a recorded baseline. <code>parameters\$ VP</code> is required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li><li>▶ <code>CompareText</code>. Captures the text of the object and compares it to a recorded baseline. <code>parameters\$ VP</code> and <code>Type</code> are required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li></ul> |





▶ ▶ ▶

| Syntax Element       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>recMethod</i> \$  | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ ID=%. The object's internal Windows ID.</li> <li>▶ ItemIndex=%. The index of the tree view item acted upon. Used only after one of these parent values: ID=%, ObjectIndex=%, Text=\$. Parent/child values are separated by a backslash and semicolons (<i>i \ ;</i>).</li> <li>▶ ItemText=\$. The text of the tree view item acted upon. Used only after one of these parent values: ID=%, ObjectIndex=%, Text=\$. Parent/child values are separated by a backslash and semicolons (<i>i \ ;</i>).</li> <li>▶ ObjectIndex=%. The number of the object among all objects of the same type in the same window.</li> <li>▶ Text=\$. The text displayed on the object.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <i>parameters</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ ExpectedResult=%. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> <li>- PASS. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail.</li> <li>- FAIL. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail.</li> </ul> </li> <li>▶ Range=&amp; , &amp;. Used with the action CompareNumeric when a numeric range comparison is being performed, as in Range=2 , 12 (test for numbers in this range). The values are inclusive.</li> <li>▶ Type=\$. Specifies the verification method to use for CompareText actions. The possible values are: CaseSensitive, CaseInsensitive, FindSubStr, FindSubStrI (case insensitive), and UserDefined. See <i>Comments</i> for more information. If UserDefined is specified, two additional parameters are required: <ul style="list-style-type: none"> <li>- DLL=\$. The full path and file name of the library that contains the function</li> <li>- Function=\$. The name of the custom function to use in comparing the text</li> </ul> </li> </ul> |

▶ ▶ ▶

## PSTreeVP

▶ ▶ ▶

| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                | <ul style="list-style-type: none"> <li>▶ <code>Value=&amp;</code>. Used with the action <code>CompareNumeric</code> when a numeric equivalence comparison is being performed, as in <code>Value=25</code> (test against the value 25).</li> <li>▶ <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points.</li> <li>▶ <code>Wait=%,%</code>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10,40</code> (retry the test every 10 seconds, but time out the test after 40 seconds).</li> </ul> |

**Comments** This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

With the `Type=$` parameter, `CaseSensitive` and `CaseInsensitive` require a full match between the current baseline text and the text captured during playback. With `FindSubStr` and `FindSubStrI`, the current baseline can be a substring of the text captured during playback. The substring can appear anywhere in the playback text. To modify the current baseline text, double-click the verification point name in the Robot Asset pane (to the left of the script).

The tree object can appear in the PeopleTools Tree Manager environment.

When you act on a particular item in a tree object, Robot uses the text of the item (plus the text of any parent items) to identify it. In the following `recMethod$` value, the tree item labeled `REPORT_VIEWS` is a child of the tree item labeled `HR_ACCESS_GROUP`.

```
"ObjectIndex=1;\;ItemText=HR_ACCESS_GROUP->REPORT_VIEWS"
```

Note the two different parent/child separators — the *backslash* (`\`) separates the window object and its child object. The *pointer* (`->`) separates the parent text item from its child text item in the tree hierarchy.

**Example** This example establishes an object data verification point for a PeopleTools tree object. The tree is identified as object 1 in the current context window.

```
Result = PSTreeVP (CompareData, "ObjectIndex=1", "VP=TREEDATA")
```

## See Also

|                |               |                |
|----------------|---------------|----------------|
| PSGrid         | PSMenuVP      | PSSpin         |
| PSGridHeader   | PSNavigator   | PSSpinVP       |
| PSGridHeaderVP | PSNavigatorVP | PSTree         |
| PSGridVP       | PSPanel       | PSTreeHeader   |
| PSMenu         | PSPanelVP     | PSTreeHeaderVP |

# PushButton

## Statement

---

**Description** Defines a custom push button.

**Syntax** Syntax A **PushButton** *x, y, dx, dy, text\$*[, *.id*]

Syntax B **Button** *x, y, dx, dy, text\$*[, *.id*]

| Syntax Element | Description                                                                                                              |
|----------------|--------------------------------------------------------------------------------------------------------------------------|
| <i>x, y</i>    | The position of the button relative to the upper left corner of the dialog box.                                          |
| <i>dx, dy</i>  | The width and height of the button.                                                                                      |
| <i>text\$</i>  | The name for the push button. If the width of this string is greater than <i>dx</i> , trailing characters are truncated. |
| <i>.id</i>     | An optional identifier used by the dialog statements that act on this control.                                           |

**Comments** A *dy* value of 14 typically accommodates text in the system font.

Use this statement to create buttons other than OK and Cancel. Use this statement in conjunction with the `ButtonGroup` statement. The two forms of the statement (`Button` and `PushButton`) are equivalent.

Use the `Button` statement only between a `Begin Dialog` and an `End Dialog` statement.

**Example** This example defines a dialog box with a combination list box and three buttons.

```

Sub main
 Dim fchoices as String
 fchoices="File1" & Chr(9) & "File2" & Chr(9) & "File3"
 Begin Dialog UserDialog 185, 94, "SQABasic Dialog Box"
 Text 9, 5, 69, 10, "Filename:", .Text1
 DropComboBox 9, 17, 88, 71, fchoices, .ComboBox1
 ButtonGroup .ButtonGroup1
 OKButton 113, 14, 54, 13
 CancelButton 113, 33, 54, 13
 PushButton 113, 57, 54, 13, "Help", .Push1
 End Dialog
 Dim mydialog as UserDialog
 On Error Resume Next
 Dialog mydialog
 If Err=102 then
 MsgBox "Dialog box canceled."
 End If
End Sub

```

## PushButton (User Action Command)

|          |              |              |                |
|----------|--------------|--------------|----------------|
| See Also | Begin Dialog | ComboBox     | OptionButton   |
|          | End Dialog   | DropComboBox | OptionGroup    |
|          | ButtonGroup  | DropListBox  | Picture        |
|          | CancelButton | GroupBox     | StaticComboBox |
|          | Caption      | ListBox      | Text           |
|          | CheckBox     | OKButton     | TextBox        |
|          |              |              |                |

## PushButton

User Action Command

▶▶SQA▶

**Description** Performs an action on a push button control.

**Syntax** **PushButton** *action%*, *recMethod\$*

| Syntax Element     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i>     | <p>The following mouse action:</p> <ul style="list-style-type: none"><li>▶ <i>MouseClick</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). Does not require coordinate information.</li></ul> <p>See Appendix E for a list of mouse click values.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <i>recMethod\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"><li>▶ <i>HTMLId=\$</i>. The text from the ID attribute of the HTML object.</li><li>▶ <i>HTMLText=\$</i>. The visible text of a Button, Reset, or Submit button of a Web page INPUT form element. The text is from the Value attribute of the INPUT tag or a BUTTON tag.</li><li>▶ <i>HTMLTitle=\$</i>. The text from the Title attribute of the HTML object.</li><li>▶ <i>ID=%</i>. The object's internal Windows ID.</li><li>▶ <i>Index=%</i>. The number of the object among all objects identified with the same base recognition method. Typically, <i>Index</i> is used after another recognition method qualifier — for example, <i>Name=\$; Index=%</i>.</li><li>▶ <i>JavaText=\$</i>. A label that identifies the object in the user interface.</li></ul> |

▶ ▶ ▶

## PushButton (User Action Command)

▶ ▶ ▶

| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                | <ul style="list-style-type: none"> <li>▶ <code>Name=\$</code>. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be <code>Command1</code>.</li> <li>▶ <code>ObjectIndex=%</code>. The number of the object among all objects of the same type in the same window.</li> <li>▶ <code>State=\$</code>. An optional qualifier for any other recognition method. There are two possible values for this setting: <code>Enabled</code> and <code>Disabled</code>. The default state is the state of the current context window (as set in the most recent <code>Window SetContext</code> command), or <code>Enabled</code> if the state has not been otherwise declared.</li> <li>▶ <code>Text=\$</code>. The text displayed on the object.</li> <li>▶ <code>Type=\$</code>. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The <code>Type</code> qualifier uses the following form: <code>Type=\$;recMethod=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>;\;</code>).</li> <li>▶ <code>VisualText=\$</code>. An optional setting used to identify an object by its visible text. It is for user clarification only and does not affect object recognition.</li> </ul> |

Comments None.

Example This example clicks the push button identified by the text OK.

```
PushButton Click, "Text=OK"
```

This example clicks the push button with the `Value` attribute `Send`. The button is located within the Web page frame named `Main`.

```
PushButton Click,
"Type=HTMLFrame;HTMLId=Main;\;Type=PushButton;HTMLText=Send"
```

See Also  
 CheckBox  
 Label  
 RadioButton

## PushButtonVP

Verification Point Command

**Description** Establishes a verification point for a push button control.

**Syntax** `Result = PushButtonVP (action%, recMethod$, parameters$)`

| Syntax Element     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i>     | <p>The type of verification to perform. Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>CompareData</code>. Captures the contents or HTML text of the object and compares it to a recorded baseline. <i>parameters\$</i> VP is required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li> <li>▶ <code>CompareNumeric</code>. Captures the numeric value of the text of the object and compares it to the value of <i>parameters\$</i> Value or Range. <i>parameters\$</i> VP and either Value or Range are required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li> <li>▶ <code>CompareProperties</code>. Captures object properties information for the object and compares it to a recorded baseline. <i>parameters\$</i> VP is required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li> <li>▶ <code>CompareText</code>. Captures the text of the object and compares it to a recorded baseline. <i>parameters\$</i> VP and <code>Type</code> are required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li> <li>▶ <code>VerifyIsBlank</code>. Checks that the object has no text. <i>parameters\$</i> VP is required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li> </ul> |
| <i>recMethod\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>HTMLId=\$</code>. The text from the ID attribute of the HTML object.</li> <li>▶ <code>HTMLText=\$</code>. The visible text of a Button, Reset, or Submit button of a Web page INPUT form element. The text is from the Value attribute of the INPUT tag or a BUTTON tag.</li> <li>▶ <code>HTMLTitle=\$</code>. The text from the Title attribute of the HTML object.</li> <li>▶ <code>ID=%</code>. The object's internal Windows ID.</li> <li>▶ <code>Index=%</code>. The number of the object among all objects identified with the same base recognition method. Typically, <code>Index</code> is used after another recognition method qualifier — for example, <code>Name=\$; Index=%</code>.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |

▶ ▶ ▶

▶ ▶ ▶

| Syntax Element            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|---------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                           | <ul style="list-style-type: none"> <li>▶ <code>JavaText=\$</code>. A label that identifies the object in the user interface.</li> <li>▶ <code>Name=\$</code>. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be <code>Command1</code>.</li> <li>▶ <code>ObjectIndex=%</code>. The number of the object among all objects of the same type in the same window.</li> <li>▶ <code>Text=\$</code>. The text displayed on the object.</li> <li>▶ <code>Type=\$</code>. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The <code>Type</code> qualifier uses the following form: <code>Type=\$;recMethod=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>;\;</code>).</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| <code>parameters\$</code> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>ExpectedResult=%</code>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> <li>- <code>PASS</code>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail.</li> <li>- <code>FAIL</code>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail.</li> </ul> </li> <li>▶ <code>Range=&amp;, &amp;</code>. Used with the action <code>CompareNumeric</code> when a numeric range comparison is being performed, as in <code>Range=2, 12</code> (test for numbers in this range). The values are inclusive.</li> <li>▶ <code>Type=\$</code>. Specifies the verification method to use for <code>CompareText</code> actions. The possible values are: <code>CaseSensitive</code>, <code>CaseInsensitive</code>, <code>FindSubStr</code>, <code>FindSubStrI</code> (case insensitive), and <code>UserDefined</code>. See <i>Comments</i> for more information. If <code>UserDefined</code> is specified, two additional parameters are required: <ul style="list-style-type: none"> <li>- <code>DLL=\$</code>. The full path and file name of the library that contains the function</li> <li>- <code>Function=\$</code>. The name of the custom function to use in comparing the text</li> </ul> </li> </ul> |

▶ ▶ ▶

## PushButtonVP

► ► ►

| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                | <ul style="list-style-type: none"><li>► <code>Value=&amp;</code>. Used with the action <code>CompareNumeric</code> when a numeric equivalence comparison is being performed, as in <code>Value=25</code> (test against the value 25).</li><li>► <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points.</li><li>► <code>Wait=%, %</code>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10, 40</code> (retry the test every 10 seconds, but time out the test after 40 seconds).</li></ul> |

**Comments** This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

With the `Type=$` parameter, `CaseSensitive` and `CaseInsensitive` require a full match between the current baseline text and the text captured during playback. With `FindSubStr` and `FindSubStrI`, the current baseline can be a substring of the text captured during playback. The substring can appear anywhere in the playback text. To modify the current baseline text, double-click the verification point name in the Robot Asset pane (to the left of the script).

**Example** This example captures the properties of the push button identified by the text `Cancel` and compares them to the recorded baseline in verification point `STBUTTON`.

```
Result=PushButtonVP(CompareProperties, "Text=Cancel", "VP=STBUTTON")
```

This example captures the data of the push button identified by the Value attribute `Clear Form`. The button is located within the Web page frame named `Main`. `PushButtonVP` compares the data to the recorded baseline in verification point `BtnData1`.

```
Result = PushButtonVP (CompareData,
 "Type=HTMLFrame;HTMLId=Main;\;Type=PushButton;HTMLText=Clear Form",
 "VP=BtnData1")
```

**See Also**

`LabelVP`  
`RadioButtonVP`



## Put Statement

---

**Description** Writes a variable to a file opened in Random or Binary mode.

**Syntax** **Put** [#]*filename%*, [*recnumber&* ], *varname*

| Syntax Element        | Description                                                                                  |
|-----------------------|----------------------------------------------------------------------------------------------|
| <i>filename%</i>      | An integer expression identifying the open file to use.                                      |
| <i>recnumber&amp;</i> | A Long expression containing the record number or the byte offset at which to start writing. |
| <i>varname</i>        | The name of the variable containing the data to write.                                       |

**Comments** *Filename%* is the number assigned to the file when it was opened. See the **Open** statement for more information.

*Recnumber&* is in the range 1 to 2,147,483,647. If *recnumber&* is omitted, the next record or byte is written.

**Note:** The commas before and after *recnumber%* are *required*, even if no *recnumber&* is specified.

*Varname* can be any variable except Object, Application Data Type or Array variables (single array elements can be used).

For Random mode, the following rules apply:

- ▶ Blocks of data are written to the file in chunks whose size is equal to the size specified in the **Len** clause of the **Open** statement. If the size of *varname* is smaller than the record length, the record is padded to the correct record size. If the size of variable is larger than the record length, an error occurs.
- ▶ For variable length **String** variables, **Put** writes two bytes of data that indicate the length of the string, then writes the string data.
- ▶ For **Variant** variables, **Put** writes two bytes of data that indicate the type of the **Variant**, then it writes the body of the **Variant** into the variable. Note that **Variants** containing strings contain two bytes of type information, followed by two bytes of length, followed by the body of the string.
- ▶ User-defined types are written as if each member were written separately, except no padding occurs between elements.

## PV

Files opened in Binary mode behave similarly to those opened in Random mode except:

- ▶ Put writes variables to the disk without record padding.
- ▶ Variable length Strings that are not part of user defined types are not preceded by the two-byte string length.

### Example

This example opens a file for Random access, puts the values 1-10 in it, prints the contents, and closes the file again.

```
Sub main
' Put the numbers 1-10 into a file
 Dim x, y
 Dim msgtext as String
 Open "C:\TEMP001" as #1
 For x=1 to 10
 Put #1,x, x
 Next x
 msgtext="The contents of the file is:" & Chr(10)
 For x=1 to 10
 Get #1,x, y
 msgtext=msgtext & y & Chr(10)
 Next x
 Close #1
 MsgBox msgtext
 Kill "C:\TEMP001"
End Sub
```

### See Also

Close      Open  
Get        Write

## PV

### Function

---

**Description** Returns the present value of a constant periodic stream of cash flows as in an annuity or a loan.

**Syntax** **PV** (*rate*, *nper*, *pmt*, *fv*, *due*)

| Syntax Element | Description                                                                                                           |
|----------------|-----------------------------------------------------------------------------------------------------------------------|
| <i>rate</i>    | Interest rate per period.                                                                                             |
| <i>nper</i>    | Total number of payment periods.                                                                                      |
| <i>pmt</i>     | Constant periodic payment per period.                                                                                 |
| <i>fv</i>      | Future value of the final lump sum amount required (in the case of a savings plan) or paid (0 in the case of a loan). |
| <i>due</i>     | An integer value for when the payments are due (0=end of each period, 1= beginning of the period).                    |

**Comments**     *Rate* is assumed constant over the life of the annuity. If payments are on a monthly schedule, then *rate* will be 0.0075 if the annual percentage rate on the annuity or loan is 9%.

**Example**        This example finds the present value of a 10-year \$25,000 annuity that will pay \$1,000 a year at 9.5%.

```

Sub main
 Dim aprate, periods
 Dim payment, annuityfv
 Dim due, presentvalue
 Dim msgtext
 aprate=9.5
 periods=120
 payment=1000
 annuityfv=25000
 Rem Assume payments are made at end of month
 due=0
 presentvalue=PV(aprate/12,periods,-payment, annuityfv,due)
 msgtext="The present value for a 10-year $25,000 annuity @ 9.5%"
 msgtext=msgtext & " with a periodic payment of $1,000 is: "
 msgtext=msgtext & Format(presentvalue, "Currency")
 MsgBox msgtext
End Sub

```

**See Also**        FV                  Pmt  
                   IPmt                PPmt  
                   IRR                  Rate  
                   NPV

## RadioButton

User Action Command



**Description**    Performs an action on an option button control.

**Syntax**            **RadioButton** *action%*, *recMethod\$*

| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                           |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i> | <p>The following mouse action:</p> <ul style="list-style-type: none"> <li>▶ <i>MouseClick</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). Does not require coordinate information.</li> </ul> <p>See Appendix E for a list of mouse click values.</p> |



## RadioButton

▶ ▶ ▶

| Syntax Element      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>recMethod</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>HTMLId=\$</code>. The text from the ID attribute of the HTML object.</li> <li>▶ <code>HTMLText=\$</code>. The visible text of a Web page INPUT form element. The text is from the Name attribute of the INPUT tag.</li> <li>▶ <code>HTMLTitle=\$</code>. The text from the Title attribute of the HTML object.</li> <li>▶ <code>ID=%</code>. The object's internal Windows ID.</li> <li>▶ <code>Index=%</code>. The number of the object among all objects identified with the same base recognition method. Typically, <code>Index</code> is used after another recognition method qualifier — for example, <code>Name=\$; Index=%</code>.</li> <li>▶ <code>JavaText=\$</code>. A label that identifies the object in the user interface.</li> <li>▶ <code>Name=\$</code>. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be <code>Command1</code>.</li> <li>▶ <code>ObjectIndex=%</code>. The number of the object among all objects of the same type in the same window.</li> <li>▶ <code>State=\$</code>. An optional qualifier for any other recognition method. There are two possible values for this setting: <code>Enabled</code> and <code>Disabled</code>. The default state is the state of the current context window (as set in the most recent <code>Window SetContext</code> command), or <code>Enabled</code> if the state has not been otherwise declared.</li> <li>▶ <code>Text=\$</code>. The text displayed on the object.</li> <li>▶ <code>Type=\$</code>. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The <code>Type</code> qualifier uses the following form: <code>Type=\$; recMethod=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>;\</code>).</li> <li>▶ <code>VisualText=\$</code>. An optional setting used to identify an object by its visible text. It is for user clarification only and does not affect object recognition.</li> </ul> |

Comments    None.

**Example** This example clicks the second option button in the window (ObjectIndex of 2).

```
RadioButton Click, "ObjectIndex=2"
```

This example clicks the option button with the Name attribute of Over 50. The option button is located within the Web page frame named Main.

```
RadioButton Click,
"Type=HTMLFrame;HTMLId=Main;\;Type=RadioButton;Name=Over 50"
```

**See Also**

CheckBox  
Label  
PushButton

## RadioButtonVP

Verification Point Command



**Description** Establishes a verification point for an option button control.

**Syntax** *Result* = **RadioButtonVP** (*action%*, *recMethod\$*, *parameters\$*)

| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i> | <p>The type of verification to perform. Valid values:</p> <ul style="list-style-type: none"> <li>▶ <b>CompareData</b>. Captures the contents or HTML text of the object and compares it to a recorded baseline. <i>parameters\$</i> VP is required; <i>ExpectedResult</i> and <i>Wait</i> are optional.</li> <li>▶ <b>CompareNumeric</b>. Captures the numeric value of the text of the object and compares it to the value of <i>parameters\$</i> Value or Range. <i>parameters\$</i> VP and either Value or Range are required; <i>ExpectedResult</i> and <i>Wait</i> are optional.</li> <li>▶ <b>CompareProperties</b>. Captures object properties information for the object and compares it to a recorded baseline. <i>parameters\$</i> VP is required; <i>ExpectedResult</i> and <i>Wait</i> are optional.</li> <li>▶ <b>CompareText</b>. Captures the text of the object and compares it to a recorded baseline. <i>parameters\$</i> VP and <i>Type</i> are required; <i>ExpectedResult</i> and <i>Wait</i> are optional.</li> <li>▶ <b>VerifyIsBlank</b>. Checks that the object has no text. <i>parameters\$</i> VP is required; <i>ExpectedResult</i> and <i>Wait</i> are optional.</li> </ul> |

▶ ▶ ▶

## RadioButtonVP

▶ ▶ ▶

| Syntax Element       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>recMethod</i> \$  | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <b>HTMLId=\$</b>. The text from the ID attribute of the HTML object.</li> <li>▶ <b>HTMLText=\$</b>. The visible text of a Web page INPUT form element. The text is from the Name attribute of the INPUT tag.</li> <li>▶ <b>HTMLTitle=\$</b>. The text from the Title attribute of the HTML object.</li> <li>▶ <b>ID=%</b>. The object's internal Windows ID.</li> <li>▶ <b>Index=%</b>. The number of the object among all objects identified with the same base recognition method. Typically, <b>Index</b> is used after another recognition method qualifier — for example, <b>Name=\$; Index=%</b>.</li> <li>▶ <b>JavaText=\$</b>. A label that identifies the object in the user interface.</li> <li>▶ <b>Name=\$</b>. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be <b>Command1</b>.</li> <li>▶ <b>ObjectIndex=%</b>. The number of the object among all objects of the same type in the same window.</li> <li>▶ <b>Text=\$</b>. The text displayed on the object.</li> <li>▶ <b>Type=\$</b>. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The <b>Type</b> qualifier uses the following form: <b>Type=\$; recMethod=\$</b>. Parent/child values are separated by a backslash and semicolons (<b>i \ i</b>).</li> </ul> |
| <i>parameters</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <b>ExpectedResult=%</b>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> <li>– <b>PASS</b>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail.</li> <li>– <b>FAIL</b>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail.</li> </ul> </li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

▶ ▶ ▶

▶ ▶ ▶

| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                | <ul style="list-style-type: none"> <li>▶ <code>Range=&amp; , &amp;</code>. Used with the action <code>CompareNumeric</code> when a numeric range comparison is being performed, as in <code>Range=2 , 12</code> (test for numbers in this range). The values are inclusive.</li> <li>▶ <code>Type=\$</code>. Specifies the verification method to use for <code>CompareText</code> actions. The possible values are: <code>CaseSensitive</code>, <code>CaseInsensitive</code>, <code>FindSubStr</code>, <code>FindSubStrI</code> (case insensitive), and <code>UserDefined</code>. See <i>Comments</i> for more information. If <code>UserDefined</code> is specified, two additional parameters are required: <ul style="list-style-type: none"> <li>- <code>DLL=\$</code>. The full path and file name of the library that contains the function</li> <li>- <code>Function=\$</code>. The name of the custom function to use in comparing the text</li> </ul> </li> <li>▶ <code>Value=&amp;</code>. Used with the action <code>CompareNumeric</code> when a numeric equivalence comparison is being performed, as in <code>Value=25</code> (test against the value 25).</li> <li>▶ <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points.</li> <li>▶ <code>Wait=% , %</code>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10 , 40</code> (retry the test every 10 seconds, but time out the test after 40 seconds).</li> </ul> |

**Comments** This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

With the `Type=$` parameter, `CaseSensitive` and `CaseInsensitive` require a full match between the current baseline text and the text captured during playback. With `FindSubStr` and `FindSubStrI`, the current baseline can be a substring of the text captured during playback. The substring can appear anywhere in the playback text. To modify the current baseline text, double-click the verification point name in the Robot Asset pane (to the left of the script).

Randomize

**Example** This example captures the text of the second option button in the window `ObjectIndex=2` and performs a case-insensitive comparison with the recorded baseline in verification point `RADIO`.

```
Result = RadioButtonVP (CompareText, "ObjectIndex=2",
"VP=RADIO;Type=CaseInsensitive")
```

This example captures the data from the option button with the `Name` attribute of `Over 50`. The option button located within the Web page frame named `Main`. `RadioButtonVP` compares the data with the recorded baseline in verification point `RadioData2`.

```
Result = RadioButtonVP (CompareData,
"Type=HTMLFrame;HTMLId=Main;\;Type=RadioButton;Name=Over 50",
"VP=RadioData2")
```

**See Also** `CheckBoxVP`  
`LabelVP`  
`PushButtonVP`

## Randomize

### Statement

---

**Description** Seeds the random number generator.

**Syntax** `Randomize [number%]`

| Syntax Element | Description                                |
|----------------|--------------------------------------------|
| <i>number%</i> | An integer value between -32768 and 32767. |

**Comments** If no *number%* argument is given, SQABasic uses the `Timer` function to initialize the random number generator.

**Example** This example generates a random string of characters using the `Randomize` statement and `Rnd` function. The second `For . . . Next` loop is to slow down processing in the first `For . . . Next` loop so that `Randomize` can be seeded with a new value each time from the `Timer` function.

```
Sub main
 Dim newline as Integer
 Dim x as Integer
 Dim y
 Dim str1 as String
 Dim str2 as String
 Dim letter as String
 Dim randomvalue
 Dim upper, lower
 Dim msgtext
```



```

upper=Asc("z")
lower=Asc("a")
newline=Chr(10)
For x=1 to 26
 Randomize timer() + x*255
 randomvalue=Int(((upper - (lower+1)) * Rnd) +lower)
 letter=Chr(randomvalue)
 str1=str1 & letter
 For y = 1 to 1500
 Next y
 Next x
 msgtext=str1
 MsgBox msgtext
End Sub

```

See Also      Rnd  
                  Timer

## Rate

### Function

---

**Description**      Returns the interest rate per period for an annuity or a loan.

**Syntax**            **Rate** (*nper*, *pmt*, *pv*, *fv*, *due*, *guess*)

| Syntax Element | Description                                                                                                              |
|----------------|--------------------------------------------------------------------------------------------------------------------------|
| <i>nper</i>    | Total number of payment periods.                                                                                         |
| <i>pmt</i>     | Constant periodic payment per period.                                                                                    |
| <i>pv</i>      | Present value of the initial lump sum amount paid (as in the case of an annuity) or received (as in the case of a loan). |
| <i>fv</i>      | Future value of the final lump sum amount required (in the case of a savings plan) or paid (0 in the case of a loan).    |
| <i>due</i>     | An integer value for when the payments are due (0=end of each period, 1= beginning of the period)                        |
| <i>guess</i>   | A ballpark estimate for the rate returned.                                                                               |

**Comments**        In general, a guess of between 0.1 (10 percent) and 0.15 (15 percent) would be a reasonable value for *guess*.

*Rate* is an iterative function: it improves the given value of *guess* over several iterations until the result is within 0.00001 percent. If it does not converge to a result within 20 iterations, it signals failure.

## Rebar

**Example** This example finds the interest rate on a 10-year \$25,000 annuity, that pays \$100 per month.

```
Sub main
 Dim aprate
 Dim periods
 Dim payment, annuitypv
 Dim annuityfv, due
 Dim guess
 Dim msgtext as String
 periods=120
 payment=100
 annuitypv=0
 annuityfv=25000
 guess=.1
 Rem Assume payments are made at end of month
 due=0
 aprate=Rate(periods,-payment,annuitypv,annuityfv, due, guess)
 aprate=(aprate*12)
 msgtext= "The percentage rate for a 10-year $25,000 annuity "
 msgtext=msgtext & "that pays $100/month has "
 msgtext=msgtext & "a rate of: " & Format(aprate, "Percent")
 MsgBox msgtext
End Sub
```

See Also

|      |      |
|------|------|
| FV   | Pmt  |
| IPmt | PPmt |
| IRR  | PV   |
| NPV  |      |

## Rebar

User Action Command



**Description** Performs an action on a rebar control.

**Syntax** **Rebar** *action%*, *recMethod\$*, *parameters\$*

| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i> | One of these mouse actions: <ul style="list-style-type: none"><li>▶ <i>MouseClicked</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> must contain <i>Coords=x,y</i>.</li><li>▶ <i>MouseDown</i>. The dragging of the mouse while mouse buttons and/or shifting keys (Ctrl, Alt, Shift) are pressed. When <i>action%</i> contains a mouse-drag value, <i>parameters\$</i> must contain <i>Coords=x1,y1,x2,y2</i>.</li></ul> See Appendix E for a list of mouse click and drag values. |



▶ ▶ ▶

| Syntax Element       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>recMethod</i> \$  | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ ID=%. The object's internal Windows ID.</li> <li>▶ ItemID=%. The application defined ID of the band. Used only after one of these parent values: ID=%, ObjectIndex=%, Name=\$, Text=\$. Parent/child values are separated by a backslash and semicolons ( ; \ ; ).</li> <li>▶ ItemIndex=%. The index of the rebar item acted upon. Used only after one of these parent values: ID=%, ObjectIndex=%, Name=\$, Text=\$. Parent/child values are separated by a backslash and semicolons ( ; \ ; ).</li> <li>▶ ItemText=\$. The text of the rebar item acted upon. Used only after one of these parent values: ID=%, ObjectIndex=%, Name=\$, Text=\$. Parent/child values are separated by a backslash and semicolons ( ; \ ; ).</li> <li>▶ Name=\$. A unique name that a developer assigns to an object to identify the object in the development environment. For example, the object name for a command button might be Command1.</li> <li>▶ ObjectIndex=%. The number of the object among all objects of the same type in the same window.</li> <li>▶ Text=\$. The text displayed on the object.</li> <li>▶ VisualText=\$. An optional setting used to identify an object by its prior label. It is for user clarification only and does not affect object recognition.</li> </ul> |
| <i>parameters</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ Coords=<i>x,y</i>. If <i>action%</i> is a mouse click, specifies the coordinates of the click, relative to the top left of the object.</li> <li>▶ Coords=<i>x1,y1,x2,y2</i>. If <i>action%</i> is a mouse drag, specifies the coordinates, where <i>x1,y1</i> are the starting coordinates of the drag, and <i>x2,y2</i> are the ending coordinates. The coordinates are relative to the top left of the object.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |

Comments None.

RebarVP

**Example** This example clicks the on the item with the text “links” in the first rebar control in the window (ObjectIndex=1) at x,y coordinates of 21,10.

```
Rebar Click, "ObjectIndex=1;\;ItemText=Links", "Coords=21,10"
```

**See Also** RebarVP

## RebarVP

Verification Point Command

»»SQA»

**Description** Establishes a verification point for a rebar control.

**Syntax** *Result* = **RebarVP** (*action%*, *recMethod\$*, *parameters\$*)

| Syntax Element      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i>      | The type of verification to perform. Valid value: <ul style="list-style-type: none"><li>▶ <b>CompareProperties</b>. Captures object properties information for the object and compares it to a recorded baseline. <i>parameters\$</i> VP is required; <b>ExpectedResult</b> and <b>Wait</b> are optional.</li></ul>                                                                                                                                                                                                                                                                                                                                                                      |
| <i>recMethod\$</i>  | Valid values: <ul style="list-style-type: none"><li>▶ <b>ID=%</b>. The object's internal Windows ID.</li><li>▶ <b>Name=\$</b>. A unique name that a developer assigns to an object to identify the object in the development environment. For example, the object name for a command button might be <b>Command1</b>.</li><li>▶ <b>ObjectIndex=%</b>. The number of the object among all objects of the same type in the same window.</li><li>▶ <b>Text=\$</b>. The text displayed on the object.</li><li>▶ <b>VisualText=\$</b>. An optional setting used to identify an object by its prior label. It is for user clarification only and does not affect object recognition.</li></ul> |
| <i>parameters\$</i> | Valid values: <ul style="list-style-type: none"><li>▶ <b>ExpectedResult=%</b>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values:<ul style="list-style-type: none"><li>– <b>PASS</b>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail.</li><li>– <b>FAIL</b>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail.</li></ul></li></ul>                                 |

▶ ▶ ▶

▶ ▶ ▶

| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                | <ul style="list-style-type: none"> <li>▶ <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points.</li> <li>▶ <code>Wait=%, %</code>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10, 40</code> (retry the test every 10 seconds, but time out the test after 40 seconds).</li> </ul> |

**Comments** This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

**Example** This example captures the properties of the first Rebar control in the window (`ObjectIndex=1`) and compares them to the recorded baseline in verification point REBAR1.

```
Result = RebarVP (CompareProperties, "ObjectIndex=1", "VP=REBAR1")
```

**See Also** Rebar

## ReDim Statement

---

**Description** Changes the upper and lower bounds of a dynamic array's dimensions.

**Syntax** **ReDim** [*Preserve*] *variableName* (*subscriptRange*, ... ) [*As* [*New*] *type*], ...

| Syntax Element        | Description                                   |
|-----------------------|-----------------------------------------------|
| <i>variableName</i>   | The variable array name to redimension.       |
| <i>subscriptRange</i> | The new upper and lower bounds for the array. |
| <i>type</i>           | The type for the data elements in the array.  |

**Comments** **ReDim** re-allocates memory for the dynamic array to support the specified dimensions, and can optionally re-initialize the array elements. **ReDim** cannot be used at the module level; it must be used inside of a procedure.

The **Preserve** option is used to change the last dimension in the array while maintaining its contents. If **Preserve** is not specified, the contents of the array are re-initialized. Numbers will be set to zero (0). Strings and variants will be set to empty (" ").

## ReDim

The *subscriptRange* is of the format:

```
[startSubscript To] endSubscript
```

If *startSubscript* is not specified, 0 is used as the default. The `Option Base` statement can be used to change the default.

A dynamic array is normally created by using `Dim` to declare an array without a specified *subscriptRange*. The maximum number of dimensions for a dynamic array created in this fashion is 8. If you need more than 8 dimensions, you can use the `ReDim` statement inside of a procedure to declare an array that has not previously been declared using `Dim` or `Global`. In this case, the maximum number of dimensions allowed is 60.

Arrays support all SQABasic data types. Arrays of arrays, dialog box records, and objects are not supported.

If the `As` clause is not used, the type of the variable can be specified by using a type character as a suffix to the name. The two different type-specification methods can be intermixed in a single `ReDim` statement (although not on the same variable).

The `ReDim` statement cannot be used to change the number of dimensions of a dynamic array once the array has been given dimensions. It can only change the upper and lower bounds of the dimensions of the array. The `LBound` and `UBound` functions can be used to query the current bounds of an array variable's dimensions.

Care should be taken to avoid redimensioning an array in a procedure that has received a reference to an element in the array in an argument; the result is unpredictable.

### Example

This example finds the net present value for a series of cash flows. The array variable that holds the cash flow amounts is initially a dynamic array that is redimensioned after the user enters the number of cash flow periods they have.

```
Sub main
 Dim aprate as Single
 Dim varray() as Double
 Dim cflowper as Integer
 Dim x as Integer
 Dim netpv as Double
 cflowper=InputBox("Enter number of cash flow periods:")
 ReDim varray(cflowper)
 For x= 1 to cflowper
 varray(x)=InputBox("Cash flow amount for period #" &x &":")
 Next x
 aprate=InputBox ("Enter discount rate:")
 If aprate>1 then aprate=aprate/100
 netpv=NPV(aprate,varray())
 MsgBox "The Net Present Value is: " & Format(netpv,"Currency")
End Sub
```

See Also      Dim            Option Base  
                  Global        Static

## RegionVP

Verification Point Command

▶▶▶SQA▶

**Description**      Establishes a verification point for a specified rectangular screen region.

**Syntax**            *Result* = **RegionVP** (*action%*, " ", *parameters\$*)

| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i> | <p>The type of verification to perform. Valid values:</p> <ul style="list-style-type: none"> <li>▶ <b>CompareImage</b>. Captures a bitmap image of the specified region on the screen and compares it to a recorded baseline. <i>parameters\$</i> <i>Coords</i> and <i>VP</i> are required; <i>ExpectedResult</i> and <i>Wait</i> are optional.</li> <li>▶ <b>WaitNegative</b>. Captures a bitmap image of the specified region on the screen and waits until it does not match the recorded baseline. <i>parameters\$</i> <i>Name</i>, <i>Wait</i>, and <i>Coords</i> are required.<br/> <b>Note:</b> Unlike <i>CompareImage</i>, this action does not use a verification point ID and does not create a failed image file if the comparison reaches timeout before failing.</li> <li>▶ <b>WaitPositive</b>. Captures a bitmap image of the specified region on the screen and waits until it matches the recorded baseline. <i>parameters\$</i> <i>Name</i>, <i>Wait</i>, and <i>Coords</i> are required.<br/> <b>Note:</b> Unlike <i>CompareImage</i>, this action does not use a verification point ID and does not create a failed image file if the comparison reaches timeout before passing.</li> </ul> |
| " "            | The second argument is always left blank.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

▶ ▶ ▶

## RegionVP

▶ ▶ ▶

| Syntax Element       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>parameters</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <b>Coords=<i>x1,y1,x2,y2</i></b>. Specifies the top-left and bottom-right screen coordinates of the region to test.</li> <li>▶ <b>ExpectedResult=<i>%</i></b>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> <li>– <b>PASS</b>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail.</li> <li>– <b>FAIL</b>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail.</li> </ul> </li> <li>▶ <b>Name=<i>\$</i></b>. For <b>WaitPositive</b> and <b>WaitNegative</b> actions, this parameter specifies the image file name to be used as a baseline in the comparison. This file is located in the same directory as the script.</li> <li>▶ <b>VP=<i>\$</i></b>. The verification point ID. IDs must be unique within a script. Required for all verification points.</li> <li>▶ <b>Wait=<i>%,%</i></b>. A Wait State that specifies the verification point's Retry value and Timeout value, as in <b>Wait=1,30</b> where 1 indicates the test is to be retried every second but timed-out after 30 seconds.</li> </ul> |

**Comments** This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

**Example** This example establishes a Region verification point identified by screen coordinates. The example compares the image to the record baseline in verification point QBMAINRG.

```
Result = RegionVP (CompareImage, "", "VP=QBMAINRG;
Coords=231,253,361,343")
```

**See Also** WindowVP



## Rem Statement

---

**Description** Identifies a line of code as a comment in an SQABasic program.

**Syntax** **Rem** *comment*

| Syntax Element | Description              |
|----------------|--------------------------|
| <i>comment</i> | The text of the comment. |

**Comments** Everything from `Rem` to the end of the line is ignored. No characters (other than spaces or tabs) can appear on the line before `Rem`.

The single quote ( `'` ) can also be used to initiate a comment. However, note that the metacommands `'CStrings`, `'$Include`, and `'$NoCStrings` are preceded by a single quote as part of their command syntax.

**Example** This example defines a dialog box with a combination list box and two buttons. The `Rem` statements describe each block of definition code.

```
Sub main
 Dim fchoices as String
 fchoices="File1" & Chr(9) & "File2" & Chr(9) & "File3"
 Begin Dialog UserDialog 185, 94, "SQABasic Dialog Box"
Rem The next two lines create the combo box
 Text 9, 5, 69, 10, "Filename:", .Text1
 DropComboBox 9, 17, 88, 71, fchoices, .ComboBox1
Rem The next two lines create the command buttons
 OKButton 113, 14, 54, 13
 CancelButton 113, 33, 54, 13
 End Dialog
 Dim mydialog as UserDialog
 On Error Resume Next
 Dialog mydialog
 If Err=102 then
 MsgBox "Dialog box canceled."
 End If
End Sub
```

**See Also** None.

Reset

## Reset

### Statement

---

**Description** Closes all open disk files and writes any data in the operating system buffers to disk.

**Syntax** **Reset**

**Comments** None.

**Example** This example creates a file, puts the numbers 1-10 in it, then attempts to Get past the end of the file. The On Error statement traps the error and execution goes to the Debugger code which uses Reset to close the file before exiting.

```
Sub main
' Put the numbers 1-10 into a file
 Dim x as Integer
 Dim y as Integer
 On Error Goto Debugger
 Open "C:\TEMP001" as #1 Len=2
 For x=1 to 10
 Put #1,x, x
 Next x
 Close #1
 msgtext="The contents of the file is:" & Chr(10)
 Open "C:\TEMP001" as #1 Len=2
 For x=1 to 10
 Get #1,x, y
 msgtext=msgtext & Chr(10) & y
 Next x
 MsgBox msgtext
done:
 Close #1
 Kill "C:\TEMP001"
 Exit Sub
Debugger:
 MsgBox "Error " & Err & " occurred. Closing open file."
 Reset
 Resume done
End Sub
```

**See Also** Close

## ResetTime

Utility Command

▶▶SQA▶

|             |                                                                                                                                                     |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| Description | Resets the delay between execution of script commands to the default delay.                                                                         |
| Syntax      | <b>ResetTime</b>                                                                                                                                    |
| Comments    | The default delay between commands is set in the Playback Options dialog box in Robot.                                                              |
| Example     | This example resets the time between execution of script commands back to the value set in the Playback Options dialog box.<br><br><b>ResetTime</b> |
| See Also    | SetTime                                                                                                                                             |

## Resume

Statement

| Description    | Halts an error-handling routine.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                |             |              |                                                                           |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|-------------|--------------|---------------------------------------------------------------------------|
| Syntax         | Syntax A <b>Resume</b> Next<br>Syntax B <b>Resume</b> <i>label</i><br>Syntax C <b>Resume</b> [ 0 ]                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                |             |              |                                                                           |
|                | <table border="1"> <thead> <tr> <th>Syntax Element</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><i>label</i></td> <td>The label that identifies the statement to go to after handling an error.</td> </tr> </tbody> </table>                                                                                                                                                                                                                                                                                                                                                                                             | Syntax Element | Description | <i>label</i> | The label that identifies the statement to go to after handling an error. |
| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                |             |              |                                                                           |
| <i>label</i>   | The label that identifies the statement to go to after handling an error.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                |             |              |                                                                           |
| Comments       | <p>When the <code>Resume Next</code> statement is used, control is passed to the statement that immediately follows the statement in which the error occurred.</p> <p>When the <code>Resume [ 0 ]</code> statement is used, control is passed to the statement in which the error occurred.</p> <p>The location of the error handler that has caught the error determines where execution will resume. If an error is trapped in the same procedure as the error handler, program execution will resume with the statement that caused the error. If an error is located in a different procedure from the error handler, program</p> |                |             |              |                                                                           |

## RichEdit

control reverts to the statement that last called out the procedure containing the error handler.

### Example

This example prints an error message if an error occurs during an attempt to open a file. The `Resume` statement jumps back into the program code at the label, `done`. From here, the program exits.

```
Sub main
 Dim msgtext, userfile
 On Error GoTo Debugger
 msgtext="Enter the filename to use:"
 userfile=InputBox$(msgtext)
 Open userfile For Input As #1
 MsgBox "File opened for input."
 ' ...etc....
 Close #1
done:
 Exit Sub
Debugger:
 msgtext="Error number " & Err & " occurred at line: " & Erl
 MsgBox msgtext
 Resume done
End Sub
```

### See Also

|                            |                                    |
|----------------------------|------------------------------------|
| <code>Erl</code>           | Error function                     |
| <code>Err</code> function  | <code>On Error</code>              |
| <code>Err</code> statement | Trappable Error Codes (Appendix B) |
| <code>Error</code>         |                                    |

## RichEdit

### User Action Command



**Description** Performs an action on a rich edit control.

**Syntax** `RichEdit` *action%*, *recMethod\$*, *parameters\$*

| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i> | One of these actions: <ul style="list-style-type: none"><li>▶ <i>MouseButtonClick</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> must contain <code>Coords=x,y</code>.</li><li>▶ <i>MouseButtonDrag</i>. The dragging of the mouse while mouse buttons and/or shifting keys (Ctrl, Alt, Shift) are pressed. When <i>action%</i> contains a mouse-drag value, <i>parameters\$</i> must contain <code>Coords=x1,y1,x2,y2</code>.</li></ul> See Appendix E for a list of mouse click and drag values. |



▶ ▶ ▶

| Syntax Element      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                 |                |             |                |                |              |            |              |           |           |
|---------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|----------------|-------------|----------------|----------------|--------------|------------|--------------|-----------|-----------|
|                     | <p>▶ <i>ScrollAction</i>. One of these scroll actions:</p> <table border="0"> <tr> <td>ScrollPageRight</td> <td>ScrollPageDown</td> </tr> <tr> <td>ScrollRight</td> <td>ScrollLineDown</td> </tr> <tr> <td>ScrollPageLeft</td> <td>ScrollPageUp</td> </tr> <tr> <td>ScrollLeft</td> <td>ScrollLineUp</td> </tr> <tr> <td>HScrollTo</td> <td>VScrollTo</td> </tr> </table> <p>HScrollTo and VScrollTo take the required parameter <i>Position=%</i>.</p> <p>If Robot cannot interpret the action being applied to a scroll bar, which happens with certain custom standalone scroll bars, it records the action as a click or drag.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | ScrollPageRight | ScrollPageDown | ScrollRight | ScrollLineDown | ScrollPageLeft | ScrollPageUp | ScrollLeft | ScrollLineUp | HScrollTo | VScrollTo |
| ScrollPageRight     | ScrollPageDown                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                 |                |             |                |                |              |            |              |           |           |
| ScrollRight         | ScrollLineDown                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                 |                |             |                |                |              |            |              |           |           |
| ScrollPageLeft      | ScrollPageUp                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                 |                |             |                |                |              |            |              |           |           |
| ScrollLeft          | ScrollLineUp                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                 |                |             |                |                |              |            |              |           |           |
| HScrollTo           | VScrollTo                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |                 |                |             |                |                |              |            |              |           |           |
| <i>recMethod\$</i>  | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <i>ID=%</i>. The object's internal Windows ID.</li> <li>▶ <i>Label=\$</i>. The text of the label object that immediately precedes the rich edit control in the Z order.</li> <li>▶ <i>Name=\$</i>. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be <i>Command1</i>.</li> <li>▶ <i>ObjectIndex=%</i>. The number of the object among all objects of the same type in the same window.</li> <li>▶ <i>State=\$</i>. An optional qualifier for any other recognition method. There are two possible values for this setting: <i>Enabled</i> and <i>Disabled</i>. The default state is the state of the current context window (as set in the most recent <i>Window SetContext</i> command), or <i>Enabled</i> if the state has not been otherwise declared.</li> <li>▶ <i>VisualText=\$</i>. An optional setting used to identify an object by its prior label. It is for user clarification only and does not affect object recognition.</li> </ul> |                 |                |             |                |                |              |            |              |           |           |
| <i>parameters\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <i>Coords=x,y</i>. If <i>action%</i> is a mouse click, specifies the <i>x,y</i> coordinates of the click, relative to the top left of the object.</li> <li>▶ <i>Coords=x1,y1,x2,y2</i>. If <i>action%</i> is a mouse drag, specifies the coordinates, where <i>x1,y1</i> are the starting coordinates of the drag, and <i>x2,y2</i> are the ending coordinates. The coordinates are relative to the top left of the object.</li> <li>▶ <i>Position=%</i>. If <i>action%</i> is <i>VScrollTo</i> or <i>HScrollTo</i>, specifies the scroll bar value of the new scrolled-to position in the scroll box. Every scroll bar has an internal range, and this value is specific to that range.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                      |                 |                |             |                |                |              |            |              |           |           |

RichEditVP

Comments None.

Example This example clicks the first rich edit control in the window `ObjectIndex=1` at `x,y` coordinates of `50,25`.

```
RichEdit Click, "ObjectIndex=1", "Coords=50,25"
```

See Also RichEditVP

## RichEditVP

Verification Point Command

»»SQA»

Description Establishes a verification point for a rich edit control.

Syntax `Result = RichEditVP (action%, recMethod$, parameters$)`

| Syntax Element           | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>action%</code>     | <p>The type of verification to perform. Valid values:</p> <ul style="list-style-type: none"><li>▶ <code>CompareNumeric</code>. Captures the numeric value of the text of the object and compares it to the value of <code>parameters\$ Value</code> or <code>Range</code>. <code>parameters\$ VP</code> and either <code>Value</code> or <code>Range</code> are required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li><li>▶ <code>CompareProperties</code>. Captures object properties information for the object and compares it to a recorded baseline. <code>parameters\$ VP</code> is required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li><li>▶ <code>CompareText</code>. Captures the text of the object and compares it to a recorded baseline. <code>parameters\$ VP</code> and <code>Type</code> are required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li><li>▶ <code>VerifyIsBlank</code>. Checks that the object has no text. <code>parameters\$ VP</code> is required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li></ul> |
| <code>recMethod\$</code> | <p>Valid values:</p> <ul style="list-style-type: none"><li>▶ <code>ID=%</code>. The object's internal Windows ID.</li><li>▶ <code>Label=\$</code>. The text of the label object that immediately precedes the rich edit control in the internal order (Z order) of windows.</li></ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |

▶ ▶ ▶

▶ ▶ ▶

| Syntax Element      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                     | <ul style="list-style-type: none"> <li>▶ Name=\$. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be Command1.</li> <li>▶ ObjectIndex=%. The number of the object among all objects of the same type in the same window.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <i>parameters\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ ExpectedResult=%. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> <li>- PASS. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail.</li> <li>- FAIL. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail.</li> </ul> </li> <li>▶ Range=&amp;, &amp;. Used with the action CompareNumeric when a numeric range comparison is being performed, as in Range=2, 12 (test for numbers in this range). The values are inclusive.</li> <li>▶ Type=\$. Specifies the verification method to use for CompareText actions. The possible values are: CaseSensitive, CaseInsensitive, FindSubStr, FindSubStrI (case insensitive), and UserDefined. See <i>Comments</i> for more information. If UserDefined is specified, two additional parameters are required: <ul style="list-style-type: none"> <li>- DLL=\$. The full path and file name of the library that contains the function</li> <li>- Function=\$. The name of the custom function to use in comparing the text</li> </ul> </li> <li>▶ Value=&amp;. Used with the action CompareNumeric when a numeric equivalence comparison is being performed, as in Value=25 (test against the value 25).</li> <li>▶ VP=\$. The verification point ID. IDs must be unique within a script. Required for all verification points.</li> <li>▶ Wait=%, %. A Wait State that specifies the verification point's Retry value and a Timeout value, as in Wait=10, 40 (retry the test every 10 seconds, but time out the test after 40 seconds).</li> </ul> |

## Right

**Comments** This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

With the `Type=$` parameter, `CaseSensitive` and `CaseInsensitive` require a full match between the current baseline text and the text captured during playback. With `FindSubStr` and `FindSubStrI`, the current baseline can be a substring of the text captured during playback. The substring can appear anywhere in the playback text. To modify the current baseline text, double-click the verification point name in the Robot Asset pane (to the left of the script).

**Example** This example captures the properties of the first rich edit control in the window `ObjectIndex=1` and compares them to the recorded baseline in verification point `TEST1A`.

```
Result=RichEditVP(CompareProperties, "ObjectIndex=1", "VP=TEST1A")
```

**See Also** `RichEdit`

## Right Function

---

**Description** Returns a string of a specified number of characters copied from the end of another string.

**Syntax** **Right**[\$](*string*%, *length*%)

| Syntax Element  | Description                                                                                                                                                    |
|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| \$              | Optional. If specified, the return type is <code>String</code> . If omitted, the function will typically return a Variant of <code>VarType 8 (String)</code> . |
| <i>string</i> % | A string or expression containing the string to copy.                                                                                                          |
| <i>length</i> % | The number of characters to copy.                                                                                                                              |

**Comments** If *length*% is greater than the length of *string*%, this function returns the whole string.

`Right` accepts any type of *string*%, including numeric values, and will convert the input value to a string. If the value of *string*% is `NULL`, a Variant of `VarType 1 (Null)` is returned.

To obtain a string of a specified number of bytes, copied from the end of another string, use `RightB`.



**Example** This example checks for the extension .BMP in a file name entered by a user and activates the Paint application if the file is found. Note this uses the `Option Compare` statement to accept either uppercase or lowercase letters for the file name extension.

```
Option Compare Text
Sub main
 Dim extension as String
 Dim filename as String
 Dim x, i
 filename=InputBox("Enter a .BMP file and path: ")
 extension=Right(filename,3)
 If extension="BMP" then
 StartApplication "pbrush.exe"
 for i = 1 to 10
 DoEvents
 next i
 AppActivate "untitled - Paint"
 DoEvents
 InputKeys "%FO" & filename & "{Enter}"
 Else
 MsgBox "File not found or extension not .BMP."
 End If
End Sub
```

**See Also**      `GetField`      `Len`      `Mid` statement  
                  `Instr`            `LTrim`            `RTrim`  
                  `Left`            `Mid` function      `Trim`

## Rmdir Statement

---

**Description** Removes a directory.

**Syntax**      **Rmdir** *path*\$

| Syntax Element | Description                                              |
|----------------|----------------------------------------------------------|
| <i>path</i> \$ | A string expression identifying the directory to remove. |

**Comments** The syntax for *path*\$ is:  
 [*drive*:][\]*directory*[\*directory*]

The *drive* argument is optional. The *directory* argument is a directory name.

The directory to be removed must be empty, except for the working ( . ) and parent ( . . ) directories.

Rnd

**Example** This example makes a new temporary directory in C:\ and then deletes it.

```
Sub main
 Dim path as String
 On Error Resume Next
 path=CurDir(C)
 If path<>"C:\" then
 ChDir "C:\"
 End If
 Mkdir "C:\TEMP01"
 If Err=75 then
 MsgBox "Directory already exists"
 Else
 MsgBox "Directory C:\TEMP01 created"
 MsgBox "Now removing directory"
 Rmdir "C:\TEMP01"
 End If
End Sub
```

**See Also** ChDir Dir  
ChDrive Mkdir  
CurDir

## Rnd

Function

---

**Description** Returns a single precision random number between 0 and 1.

**Syntax** **Rnd** [(*number!*)]

| Syntax Element | Description                                                                                                                                                                          |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>number!</i> | A numeric expression to specify how to generate the random numbers. (<0=use the number specified, >0=use the next number in the sequence, 0=use the number most recently generated.) |

**Comments** If *number!* is omitted, Rnd uses the next number in the sequence to generate a random number. The same sequence of random numbers is generated whenever Rnd is run, unless the random number generator is re-initialized by the Randomize statement.

**Example** This example generates a random string of characters within a range. The `Rnd` function is used to set the range between lowercase a and z. The second `For...Next` loop slows down processing in the first `For...Next` loop so that `Randomize` can be seeded with a new value each time from the `Timer` function.

```
Sub main
 Dim x as Integer
 Dim y
 Dim str1 as String
 Dim str2 as String
 Dim letter as String
 Dim randomvalue
 Dim upper, lower
 Dim msgtext
 Dim newline as Integer
 upper=Asc("z")
 lower=Asc("a")
 newline=Chr(10)
 For x=1 to 26
 Randomize timer() + x*255
 randomvalue=Int(((upper - (lower+1)) * Rnd) +lower)
 letter=Chr(randomvalue)
 str1=str1 & letter
 For y = 1 to 1500
 Next y
 Next x
 msgtext=str1
 MsgBox msgtext
 End Sub
```

**See Also**      `Exp`      `Log`                      `Sqr`  
                  `Fix`      `Randomize`  
                  `Int`      `Sgn`

## Rset

### Statement

---

**Description**      Right aligns one string inside another string.

**Syntax**              **Rset** *string\$* = *string-expression*

| Syntax Element           | Description                                                        |
|--------------------------|--------------------------------------------------------------------|
| <i>string\$</i>          | The string to contain the right-aligned characters.                |
| <i>string-expression</i> | The string containing the characters to put into <i>string\$</i> . |

RTrim

**Comments** If *string\$* is longer than *string-expression*, the leftmost characters of *string\$* are replaced with spaces.

If *string\$* is shorter than *string-expression*, only the leftmost characters of *string-expression* are copied.

Rset cannot be used to assign variables of different user-defined types.

**Example** This example uses Rset to right-align an amount entered by the user in a field that is 15 characters long. It then pads the extra spaces with asterisks ( \* ) and adds a dollar sign ( \$ ) and decimal places (if necessary).

```
Sub main
 Dim amount as String*15
 Dim x
 Dim msgtext
 Dim replacement
 Dim position as Integer
 Dim length as Integer
 replacement="*"
 amount=InputBox("Enter an amount:")
 position=InStr(amount, ".")
 If Right(amount,3)<>".00" then
 amount=Rtrim(amount) & ".00"
 End If
 Rset amount="$" & Rtrim(amount)
 length=15-Len(Ltrim(amount))
 For x=1 to length
 Mid(amount,x)=replacement
 Next x
 MsgBox "Formatted amount: " & amount
End Sub
```

**See Also** Lset

## RTrim Function

---

**Description** Copies a string and removes any trailing spaces.

**Syntax** **RTrim**[\$](*expression*)

| Syntax Element    | Description                                                                                                                             |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| \$                | Optional. If specified the return type is String. If omitted the function will typically return a Variant of VarType 8 (string).        |
| <i>expression</i> | The expression to trim. The expression can be a string, or it can be a numeric data type which Robot passes to the command as a string. |

**Comments** If the value of *string\$* is NULL, a Variant of VarType 1 (Null) is returned.

**Example** This example asks for an amount and then right-aligns it in a field that is 15 characters long. It uses `Rtrim` to trim any trailing spaces in the amount string, if the number entered by the user is less than 15 digits.

```

Sub main
 Dim position as Integer
 Dim length as Integer
 Dim amount as String*15
 Dim x, msgtext, replacement
 replacement="X"
 amount=InputBox("Enter an amount:")
 position=InStr(amount, ".")
 If position=0 then
 amount=Rtrim(amount) & ".00"
 End If
 Rset amount="$" & Rtrim(amount)
 length=15-Len(Ltrim(amount))
 For x=1 to length
 Mid(amount,x)=replacement
 Next x
 MsgBox "Formatted amount: " & amount
End Sub

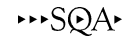
```

**See Also**

|          |               |
|----------|---------------|
| GetField | Mid function  |
| Left     | Mid statement |
| Len      | Right         |
| LTrim    | Trim          |

## ScrollBar

User Action Command



**Description** Performs an action on a scroll bar.

**Syntax** `ScrollBar action%, recMethod$, parameters$`

| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i> | <p>One of these actions:</p> <ul style="list-style-type: none"> <li>▶ <i>MouseClicked</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> must contain <code>Coords=x,y</code>.</li> <li>▶ <i>MouseDown</i>. The dragging of the mouse while mouse buttons and/or shifting keys (Ctrl, Alt, Shift) are pressed. When <i>action%</i> contains a mouse-drag value, <i>parameters\$</i> must contain <code>Coords=x1,y1,x2,y2</code>.</li> </ul> <p>See Appendix E for a list of mouse click and drag values.</p> |



## ScrollBar

▶ ▶ ▶

| Syntax Element      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                 |                |             |                |                |              |            |              |           |           |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|----------------|-------------|----------------|----------------|--------------|------------|--------------|-----------|-----------|
|                     | <p>▶ <i>ScrollAction</i>. One of these scroll actions:</p> <table border="0"> <tr> <td>ScrollPageRight</td> <td>ScrollPageDown</td> </tr> <tr> <td>ScrollRight</td> <td>ScrollLineDown</td> </tr> <tr> <td>ScrollPageLeft</td> <td>ScrollPageUp</td> </tr> <tr> <td>ScrollLeft</td> <td>ScrollLineUp</td> </tr> <tr> <td>HScrollTo</td> <td>VScrollTo</td> </tr> </table> <p>HScrollTo and VScrollTo take the required parameter <code>Position=%</code>.</p> <p>If Robot cannot interpret the action being applied to a scroll bar, which happens with certain custom standalone scroll bars, it records the action as a click or drag.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | ScrollPageRight | ScrollPageDown | ScrollRight | ScrollLineDown | ScrollPageLeft | ScrollPageUp | ScrollLeft | ScrollLineUp | HScrollTo | VScrollTo |
| ScrollPageRight     | ScrollPageDown                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                 |                |             |                |                |              |            |              |           |           |
| ScrollRight         | ScrollLineDown                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                 |                |             |                |                |              |            |              |           |           |
| ScrollPageLeft      | ScrollPageUp                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                 |                |             |                |                |              |            |              |           |           |
| ScrollLeft          | ScrollLineUp                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                 |                |             |                |                |              |            |              |           |           |
| HScrollTo           | VScrollTo                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                 |                |             |                |                |              |            |              |           |           |
| <i>recMethod</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>ID=%</code>. The object's internal Windows ID.</li> <li>▶ <code>Index=%</code>. The number of the object among all objects identified with the same base recognition method. Typically, <code>Index</code> is used after another recognition method qualifier — for example, <code>Name=\$; Index=%</code>.</li> <li>▶ <code>JavaText=\$</code>. A label that identifies the object in the user interface.</li> <li>▶ <code>Name=\$</code>. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be <code>Command1</code>.</li> <li>▶ <code>ObjectIndex=%</code>. The number of the object among all objects of the same type in the same window.</li> <li>▶ <code>State=\$</code>. An optional qualifier for any other recognition method. There are two possible values for this setting: <code>Enabled</code> and <code>Disabled</code>. The default state is the state of the current context window (as set in the most recent <code>Window SetContext</code> command), or <code>Enabled</code> if the state has not been otherwise declared.</li> <li>▶ <code>Type=\$</code>. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The <code>Type</code> qualifier uses the following form: <code>Type=\$; recMethod=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>; \ ;</code>).</li> </ul> |                 |                |             |                |                |              |            |              |           |           |

▶ ▶ ▶



| Syntax Element      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>parameters\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <i>Coords=x,y</i>. If <i>action%</i> is a mouse click, specifies the coordinates of the click, relative to the top left of the object.</li> <li>▶ <i>Coords=x1,y1,x2,y2</i>. If <i>action%</i> is a mouse drag, specifies the coordinates, where <i>x1,y1</i> are the starting coordinates of the drag, and <i>x2,y2</i> are the ending coordinates. The coordinates are relative to the top left of the object.</li> <li>▶ <i>Position=%</i>. If <i>action%</i> is <i>VScrollTo</i> or <i>HScrollTo</i>, specifies the scroll bar value of the new scrolled-to position. Every scroll bar has an internal range, and this parameter value is specific to that range.</li> </ul> |

Comments None.

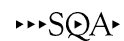
Example This example moves the thumb of the first scroll bar in the window (ObjectIndex=1) to the 159th position.

```
ScrollBar HScrollTo, "ObjectIndex=1", "Position=159"
```

See Also  
 ComboBox  
 ComboBox  
 ListBox

## ScrollBarVP

Verification Point Command



Description Establishes a verification point for a scroll bar.

Syntax *Result* = **ScrollBarVP** (*action%*, *recMethod\$*, *parameters\$*)

| Syntax Element | Description                                                                                                                                                                                                                                                                                                                  |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i> | <p>The type of verification to perform. Valid value:</p> <ul style="list-style-type: none"> <li>▶ <i>CompareProperties</i>. Captures object properties information for the object and compares it to a recorded baseline. <i>parameters\$ VP</i> is required; <i>ExpectedResult</i> and <i>Wait</i> are optional.</li> </ul> |



ScrollBarVP

▶ ▶ ▶

| Syntax Element       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>recMethod</i> \$  | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <i>ID</i>=%. The object's internal Windows ID.</li> <li>▶ <i>Index</i>=%. The number of the object among all objects identified with the same base recognition method. Typically, <i>Index</i> is used after another recognition method qualifier — for example, <i>Name</i>=\$ ; <i>Index</i>=%.</li> <li>▶ <i>JavaText</i>=\$. A label that identifies the object in the user interface.</li> <li>▶ <i>Name</i>=\$. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be <i>Command1</i>.</li> <li>▶ <i>ObjectIndex</i>=%. The number of the object among all objects of the same type in the same window.</li> <li>▶ <i>Type</i>=\$. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The <i>Type</i> qualifier uses the following form: <i>Type</i>=\$ ; <i>recMethod</i>=\$. Parent/child values are separated by a backslash and semicolons (<i>;</i> \ <i>;</i>).</li> </ul> |
| <i>parameters</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <i>ExpectedResult</i>=%. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> <li>– <i>PASS</i>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail.</li> <li>– <i>FAIL</i>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail.</li> </ul> </li> <li>▶ <i>VP</i>=\$. The verification point ID. IDs must be unique within a script. Required for all verification points.</li> <li>▶ <i>Wait</i>=% , %. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <i>Wait</i>=10 , 40 (retry the test every 10 seconds, but time out the test after 40 seconds).</li> </ul>                                                                                                            |

Comments This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.



**Example** This example captures the properties of the first scroll bar control in the window `ObjectIndex=1` and compares them to the recorded baseline in verification point `SCROLLBAR`.

```
Result=ScrollBarVP(CompareProperties,"ObjectIndex=1","VP=SCROLLBAR")
```

**See Also** `ComboBoxVP`  
`ComboBoxVP`  
`ListBoxVP`

## Second Function

---

**Description** Returns the second component (0-59) of a date-time value.

**Syntax** `Second(time)`

| Syntax Element | Description                                 |
|----------------|---------------------------------------------|
| <i>time</i>    | An expression containing a date time value. |

**Comments** `Second` accepts any type of *time* including strings and will attempt to convert the input value to a date value.

The return value is a Variant of VarType 2 (integer). If the value of *time* is NULL, a Variant of VarType 1 (Null) is returned.

**Example** This example displays the last saved date and time for a file whose name is entered by the user.

```
Sub main
 Dim filename as String
 Dim ftime
 Dim hr, min
 Dim sec
 Dim msgtext as String
i: msgtext="Enter a filename:"
 filename=InputBox(msgtext)
 If filename="" then Exit Sub
 On Error Resume Next
 ftime=FileDateTime(filename)
 If Err<>0 then
 MsgBox "Error in file name. Try again."
 Goto i:
 End If
 hr=Hour(ftime)
 min=Minute(ftime)
 sec=Second(ftime)
 MsgBox "The file's time is: " & hr & ":" & min & ":" & sec
End Sub
```

## Seek (Function)

|          |        |                |
|----------|--------|----------------|
| See Also | Day    | Time function  |
|          | Hour   | Time statement |
|          | Minute | Weekday        |
|          | Month  | Year           |
|          | Now    |                |

## Seek Function

---

**Description** Returns the current file position for an open file.

**Syntax** **Seek**(*filenumber%*)

| Syntax Element     | Description                                              |
|--------------------|----------------------------------------------------------|
| <i>filenumber%</i> | An integer expression identifying an open file to query. |

**Comments** *filenumber%* is the number assigned to the file when it was opened. See the **Open** statement for more information.

For files opened in **Random** mode, **Seek** returns the number of the next record to be read or written. For all other modes, **Seek** returns the file offset for the next operation. The first byte in the file is at offset 1, the second byte is at offset 2, and so on. The return value is a **Long**.

**Example** This example reads the contents of a sequential file line by line (to a carriage return) and displays the results. The second sub procedure, **CREATEFILE**, creates the file **C:\TEMP001** used by the main sub procedure.

```
Declare Sub createfile
Sub main
 Dim msgtext as String
 Dim testscore as String
 Dim x, y, newline
 Call createfile
 Open "C:\TEMP001" for Input as #1
 x=1
 newline=Chr(10)
 msgtext= "The test scores are: " & newline
 Do Until x=Lof(1)
 Line Input #1, testscore
 x=x+1
 y=Seek(1)
 If y>Lof(1) then
 x=Lof(1)
 Else
 Seek 1,y
 End If
 msgtext=msgtext & newline & testscore
 Loop
```

```

MsgBox msgtext
Close #1
Kill "C:\TEMP001"
End Sub

Sub createfile()
Rem Put the numbers 10-100 into a file
Dim x as Integer
Open "C:\TEMP001" for Output as #1
For x=10 to 100 step 10
 Write #1, x
Next x
Close #1
End Sub

```

See Also      Get            Put  
                   Open            Seek statement

## Seek Statement

---

**Description**    Sets the position within an open file for the next read or write operation.

**Syntax**            **Seek** [#]*filenumber%*, *position&*

| Syntax Element       | Description                                                                                                        |
|----------------------|--------------------------------------------------------------------------------------------------------------------|
| <i>filenumber%</i>   | An integer expression identifying an open file to query.                                                           |
| <i>position&amp;</i> | A numeric expression for the starting position of the next read or write operation (record number or byte offset). |

**Comments**        The *Seek* statement. If you write to a file after seeking beyond the end of the file, the file's length is extended. SQABasic will return an error message if a *Seek* operation is attempted that specifies a negative or zero position.

*Filenumber%* is an integer expression identifying the open file to *Seek* in. See the *Open* statement for more details.

For files opened in *Random* mode, *position&* is a record number; for all other modes, *position&* is a byte offset. *Position&* is in the range 1 to 2,147,483,647. The first byte or record in the file is at position 1, the second is at position 2, and so on.

## Seek (Statement)

**Example** This example reads the contents of a sequential file line by line (to a carriage return) and displays the results. The second sub procedure, CREATEFILE, creates the file C:\TEMP001 used by the main sub procedure.

```
Declare Sub createfile
Sub main
 Dim msgtext as String
 Dim testscore as String
 Dim x
 Dim y
 Dim newline
 Call createfile
 Open "C:\TEMP001" for Input as #1
 x=1
 newline=Chr(10)
 msgtext= "The test scores are: " & newline
 Do Until x=Lof(1)
 Line Input #1, testscore
 x=x+1
 y=Seek(1)
 If y>Lof(1) then
 x=Lof(1)
 Else
 Seek 1,y
 End If
 msgtext=msgtext & newline & testscore
 Loop
 MsgBox msgtext
 Close #1
 Kill "C:\TEMP001"
End Sub

Sub createfile()
 Rem Put the numbers 10-100 into a file
 Dim x as Integer
 Open "C:\TEMP001" for Output as #1
 For x=10 to 100 step 10
 Write #1, x
 Next x
 Close #1
End Sub
```

**See Also**

|      |               |
|------|---------------|
| Get  | Put           |
| Open | Seek function |

## Select Case Statement

---

**Description** Executes a series of statements, depending on the value of an expression.

**Syntax**

```
Select Case testexpression
 [Case expressionlist
 [statement_block]]
 [Case expressionlist
 [statement_block]]
 [Case Else
 [statement_block]]
End Select
```

| Syntax Element         | Description                                                                       |
|------------------------|-----------------------------------------------------------------------------------|
| <i>testexpression</i>  | Any expression containing a variable to test.                                     |
| <i>expressionlist</i>  | One or more expressions that contain a possible value for <i>testexpression</i> . |
| <i>statement_block</i> | The statements to execute if <i>testexpression</i> equals <i>expressionlist</i> . |

**Comments** When there is a match between *testexpression* and one of the values in *expressionlist*, the *statement\_block* following the Case clause is executed. When the next Case clause is reached, execution control goes to the statement following the End Select statement.

The *expressionlist(s)* can be a comma-separated list of expressions of the following forms:

```
expression
expression To expression
Is comparison_operator expression
```

The type of each *expression* must be compatible with the type of *testexpression*.

Note that when the To keyword is used to specify a range of values, the smaller value must appear first. The *comparison\_operator* used with the Is keyword is one of: <, >, =, <=, >=, <>.

Each *statement\_block* can contain any number of statements on any number of lines.

## Select Case

### Example

This example tests the attributes for a file and if it is hidden, changes it to a non-hidden file.

```
Sub main

 Dim filename as String
 Dim attribs, saveattribs as Integer
 Dim answer as Integer
 Dim archno as Integer
 Dim msgtext as String

 archno=32
 On Error Resume Next
 msgtext="Enter name of a file:"
 filename=InputBox(msgtext)
 attribs=GetAttr(filename)

 If Err<>0 then
 MsgBox "Error in filename. Re-run Program."
 Exit Sub
 End If

 saveattribs=attribs

 If attribs>= archno then
 attribs=attribs-archno
 End If

 Select Case attribs

 Case 2,3,6,7
 msgtext=" File: " &filename & " is hidden." & Chr(10)
 msgtext=msgtext & Chr(10) & " Change it?"
 answer=MsgBox(msgtext,308)
 If answer=6 then
 SetAttr filename, saveattribs-2
 MsgBox "File is no longer hidden."
 Exit Sub
 End If
 MsgBox "Hidden file not changed."

 Case Else
 MsgBox "File was not hidden."
 End Select

End Sub
```

### See Also

```
If...Then...Else
On...Goto
Option Compare
```

## SendKeys Statement

---

This command should no longer be used. Use the `InputKeys` command instead. To maintain the upward compatibility of your existing scripts, the command does not cause an error.

## Set Statement

---

**Description** Assigns a variable to an OLE2 object.

**Syntax** `Set variableName = expression`

| Syntax Element            | Description                                                                                                  |
|---------------------------|--------------------------------------------------------------------------------------------------------------|
| <code>variableName</code> | An object variable or a Variant variable.                                                                    |
| <code>expression</code>   | An expression that evaluates to an object--typically a function, an object member, or <code>Nothing</code> . |

**Comments** The following example shows the syntax for the `Set` statement:

```
Dim OLE2 As Object
Set OLE2 = CreateObject("spoly.cpoly")
OLE2.reset
```

**Note:** If you omit the keyword `Set` when assigning an object variable, `SQABasic` will try to copy the default member of one object to the default member of another. This usually results in a runtime error:

```
' Incorrect code - tries to copy default member!
OLE2 = GetObject(, "spoly.cpoly")
```

`Set` differs from `Let` in that `Let` assigns an expression to an `SQABasic` variable. For example,

```
Set o1 = o2 'Sets the object reference
Let o1 = o2 'Sets the value of the default member
```

## SetAttr

### Example

This example displays a list of open files in the software application, VISIO. It uses the Set statement to assign VISIO and its document files to object variables. To see how this example works, you need to start VISIO and open one or more documents.

```
Sub main

 Dim visio as Object
 Dim doc as Object
 Dim msgtext as String
 Dim i as Integer, doccount as Integer

 'Initialize Visio
 Set visio = GetObject("visio.application") ' find Visio
 If (visio Is Nothing) then
 MsgBox "Couldn't find Visio!"
 Exit Sub
 End If

 'Get # of open Visio files
 doccount = visio.documents.count 'OLE2 call to Visio
 If doccount=0 then
 msgtext="No open Visio documents."
 Else
 msgtext="The open files are: " & Chr$(13)
 For i = 1 to doccount
 ' access Visio's document method
 Set doc = visio.documents(i)
 msgtext=msgtext & Chr$(13) & doc.name
 Next i
 End If
 MsgBox msgtext

End Sub
```

### See Also

|              |              |
|--------------|--------------|
| Class List   | Nothing      |
| CreateObject | Object Class |
| Is           | Typeof       |
| New          |              |

## SetAttr

### Statement

---

**Description** Sets the attributes for a file.

**Syntax** **SetAttr** *pathname\$, attributes%*



| Syntax Element     | Description                                                                                                                                                                                                                                                                          |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>pathname\$</i>  | A string expression containing the file name to modify.                                                                                                                                                                                                                              |
| <i>attributes%</i> | An integer containing the new attributes for the file.<br>Valid attributes:<br><ul style="list-style-type: none"> <li>0. Normal file</li> <li>1. Read-only file</li> <li>2. Hidden file</li> <li>4. System file</li> <li>32. Archive - file has changed since last backup</li> </ul> |

**Comments** Wildcards are not allowed in *pathname\$*. If the file is open, you can modify its attributes, but only if it is opened for Read access.

**Example** This example tests the attributes for a file and if it is hidden, changes it to a normal (not hidden) file.

```

Sub main
 Dim filename as String
 Dim attribs, saveattribs as Integer
 Dim answer as Integer
 Dim archno as Integer
 Dim msgtext as String
 archno=32
 On Error Resume Next
 msgtext="Enter name of a file:"
 filename=InputBox(msgtext)
 attribs=GetAttr(filename)
 If Err<>0 then
 MsgBox "Error in filename. Re-run Program."
 Exit Sub
 End If
 saveattribs=attribs
 If attribs>= archno then
 attribs=attribs-archno
 End If
 Select Case attribs
 Case 2,3,6,7
 msgtext=" File: " &filename & " is hidden." & Chr(10)
 msgtext=msgtext & Chr(10) & " Change it?"
 answer=MsgBox(msgtext,308)
 If answer=6 then
 SetAttr filename, saveattribs-2
 MsgBox "File is no longer hidden."
 Exit Sub
 End If
 MsgBox "Hidden file not changed."
 Case Else
 MsgBox "File was not hidden."
 End Select
End Sub

```

**See Also** FileAttr  
GetAttr

## SetField

Function

»»SQ«

**Description** Replaces a field within a string and returns the modified string. .

**Syntax** **SetField**[\$](*string*\$, *field\_number*%, *field*\$, *separator\_chars*\$ )

| Syntax Element           | Description                                                                                                                                                |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| \$                       | Optional. If specified, the return type is <code>String</code> . If omitted, the function typically returns a Variant of <code>VarType 8 (String)</code> . |
| <i>string</i> \$         | A string consisting of a series of fields, separated by <i>separator_char</i> \$.                                                                          |
| <i>field_number</i> %    | An integer for the field to replace within <i>string</i> \$.                                                                                               |
| <i>field</i> \$          | An expression containing the new value for the field.                                                                                                      |
| <i>separator_char</i> \$ | A string containing the character(s) used to separate the fields in <i>string</i> \$.                                                                      |

**Comments** *separator\_char*\$ can contain multiple separator characters, although the first one will be used as the separator character.

The *field\_number*% starts with 1. If *field\_number*% is greater than the number of fields in the string, the returned string will be extended with separator characters to produce a string with the proper number of fields.

It is legal for the new *field*\$ value to be a different size than the old value.

**Example** This example extracts the last name from a full name entered by the user.

```
Sub main
 Dim username as String
 Dim position as Integer
 username=InputBox("Enter your full name:")
 Do
 position=InStr(username," ")
 If position=0 then
 Exit Do
 End If
 username=SetField(username,1," "," ")
 username=Ltrim(username)
 Loop
 MsgBox "Your last name is: " & username
End Sub
```

**See Also** `GetField`

## SetProcID

Flow Control Command

▶▶SQA▶

This command is obsolete in the current version of SQABasic and should no longer be used. To maintain the upward compatibility of your existing scripts, the command does not cause an error, but it has no effect on script execution.

## SetThinkAvg

Timing and Coordination Command

▶▶SQA▶

**Description** Sets the average “think time” delay for the next user action.

**Syntax** **SetThinkAvg** *avgThinkTime%*

| Syntax Element       | Description                                                                                                                                                                                                                                                                |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>avgThinkTime%</i> | The delay, in milliseconds Robot observed between two actions during recording. During playback in Robot, Robot used <i>avgThinkTime</i> as the actual think time delay. During playback in LoadTest, LoadTest uses <i>avgThinkTime</i> to calculate the think time delay. |

**Comments** Robot records this command if Record Think Time is selected in the General tab of the GUI Record Options dialog box. During playback, Robot performs think time delays only if Use recorded think time is selected in the Playback tab of the GUI Playback Options dialog box.

Robot delays execution of the next user action during synchronized testing. If running in a LoadTest Schedule, LoadTest may adjust the *AvgThinkTime%* to prevent playback of simultaneous GUI agents running in lock step. If running as a stand-alone Robot script, the *avgThinkTime%* is the actual delay time.

**Example** This example sets an average think time of 1500 milliseconds (1.5 seconds).

```
SetThinkAvg 1500
```

**See Also** TypingDelays

## SetTime

Utility Command

»»SQA»

**Description** Sets the delay between script commands to the specified number of millisecond.

**Syntax** `SetTime (TimeInterval&)`

| Syntax Element           | Description                                          |
|--------------------------|------------------------------------------------------|
| <i>TimeInterval&amp;</i> | The number of millisecond to delay between commands. |

**Comments** This command overrides the Delay Between Commands setting in the Playback Options dialog box in Robot.

**Example** This example sets the delay between execution of script commands to 1000 milliseconds (1 second).

```
SetTime (1000)
```

**See Also** ResetTime

## Sgn

Function

**Description** Returns a value indicating the sign of a number.

**Syntax** `Sgn (number)`

| Syntax Element | Description                          |
|----------------|--------------------------------------|
| <i>number</i>  | An expression for the number to use. |

**Comments** The value that the Sgn function returns depends on the sign of *number*:

- ▶ For *numbers* > 0, Sgn (*number*) returns 1.
- ▶ For *numbers* = 0, Sgn (*number*) returns 0.
- ▶ For *numbers* < 0, Sgn (*number*) returns -1.

**Example** This example tests the value of the variable profit and displays 0 for profit if it is a negative number. The sub procedure uses Sgn to determine whether profit is positive, negative or zero.

```

Sub main
 Dim profit as Single
 Dim expenses, sales
 expenses=InputBox("Enter total expenses: ")
 sales=InputBox("Enter total sales: ")
 profit=Val(sales)-Val(expenses)
 If Sgn(profit)=1 then
 MsgBox "Yeah! We turned a profit!"
 ElseIf Sgn(profit)=0 then
 MsgBox "Okay. We broke even."
 Else
 MsgBox "Uh, oh. We lost money."
 End If
End Sub

```

See Also

|     |     |
|-----|-----|
| Exp | Log |
| Fix | Rnd |
| Int | Sqr |

## Shell

### Function

---

**Description** Starts a Windows application and returns its task ID.

**Syntax** **Shell**(*pathname\$*, [*windowstyle%*])

| Syntax Element      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|---------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>pathname\$</i>   | The name of the program to execute.                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <i>windowstyle%</i> | An integer value for the style of the program's window (1-7). <i>Windowstyle%</i> is one of the following values: <ol style="list-style-type: none"> <li>1. Normal window with focus</li> <li>2. Minimized with focus</li> <li>3. Maximized with focus</li> <li>4. Normal window without focus</li> <li>7. Minimized without focus</li> </ol> If <i>windowstyle%</i> is not specified, the default of <i>windowstyle%</i> = 1 is assumed (normal window with focus). |

**Comments** Shell runs an executable program. *Pathname\$* can be the name of any valid .COM, .EXE, .BAT, or .PIF file. Arguments or command line switches can be included. If *pathname\$* is not a valid executable file name, or if Shell cannot start the program, an error message occurs.

Shell returns the task ID for the program, a unique number that identifies the running program.

Sin

**Example** This example runs Notepad in maximized format.

```
Sub main
 Shell "Notepad.exe",3
 InputKeys "Hello, world.{enter}Notepad is maximized."
End sub
```

**See Also** AppActivate  
Command  
InputKeys

## Sin Function

---

**Description** Returns the sine of an angle specified in radians.

**Syntax** **Sin**(*number*)

| Syntax Element | Description                                    |
|----------------|------------------------------------------------|
| <i>number</i>  | An expression containing the angle in radians. |

**Comments** The return value will be between -1 and 1. The return value is single-precision if the angle is an integer, currency or single-precision value, double precision for a long, Variant or double-precision value. The angle is specified in radians, and can be either positive or negative.

To convert degrees to radians, multiply by (PI/180). The value of PI is 3.14159.

**Example** This example finds the height of the building, given the length of a roof and the roof pitch.

```
Sub main
 Dim height, rooflength
 Dim pitch
 Dim msgtext
 Const PI=3.14159
 Const conversion= PI/180
 pitch=InputBox("Enter the roof pitch in degrees:")
 pitch=pitch*conversion
 rooflength=InputBox("Enter the length of the roof in feet:")
 height=Sin(pitch)*rooflength
 msgtext="The height of the building is "
 msgtext=msgtext & Format(height, "##.##") & " feet."
 MsgBox msgtext
End Sub
```

See Also     Atn  
               Cos  
               Tan  
               Derived Trigonometric functions (Appendix D)

## Space Function

---

Description   Returns a string of spaces.

Syntax       **Space** [ \$ ] ( *number* )

| Syntax Element | Description                                                                                                                                         |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| \$             | Optional. If specified the return type is <code>String</code> . If omitted, the function will return a <code>Variant of VarType 8 (String)</code> . |
| <i>number</i>  | A numeric expression for the number of spaces to return.                                                                                            |

Comments     *number* can be any numeric data type, but will be rounded to an integer. *number* must be between 0 and 32,767.

Example       This example prints the octal numbers from 1 to 15 as a two-column list and uses `Space` to separate the columns.

```
Sub main
 Dim x,y
 Dim msgtext
 Dim nofspace
 msgtext="Octal numbers from 1 to 15:" & Chr(10)
 For x=1 to 15
 nofspace=10
 y=Oct(x)
 If Len(x)=2 then
 nofspace=nofspace-2
 End If
 msgtext=msgtext & Chr(10) & x & Space(nospace) & y
 Next x
 MsgBox msgtext
End Sub
```

See Also     Spc  
               String

Spc

## Spc Function

---

Description Prints a number of spaces.

Syntax **Spc** ( *n* )

| Syntax Element | Description                                    |
|----------------|------------------------------------------------|
| <i>n</i>       | An integer for the number of spaces to output. |

Comments The SPC function can be used only inside Print statement.

When the Print statement is used, the SPC function will use the following rules for determining the number of spaces to output:

1. If *n* is less than the total line width, SPC outputs *n* spaces.
2. If *n* is greater than the total line width, SPC outputs *n* Mod *width* spaces.
3. If the difference between the current print position and the output line width (call this difference *x*) is less than *n* or *n* Mod *width*, then SPC skips to the next line and outputs *n* - *x* spaces.

To set the width of a print line, use the Width statement.

Example This example puts five spaces and the string ABCD to a file. The five spaces are derived by taking 15 MOD 10, or the remainder of dividing 15 by 10.

```
Sub main
 Dim str1 as String
 Dim x as String*10
 str1="ABCD"
 Open "C:\TEMP001" For Output As #1
 Width #1, 10
 Print #1, Spc(15); str1
 Close #1
 Open "C:\TEMP001" as #1 Len=12
 Get #1, 1,x
 MsgBox "The contents of the file is: " & x
 Close #1
 Kill "C:\TEMP001"
End Sub
```

See Also Print Tab  
Space Width



# SpinControl

User Action Command

»»SQA»

Description Performs an action on a spin control.

Syntax **SpinControl** *action%*, *recMethod\$*, *parameters\$*

| Syntax Element     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i>     | <p>One of these mouse actions:</p> <ul style="list-style-type: none"> <li>▶ <i>MouseClicked</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> must contain <i>Coords=x,y</i>.</li> <li>▶ <i>MouseDown</i>. The dragging of the mouse while mouse buttons and/or shifting keys (Ctrl, Alt, Shift) are pressed. When <i>action%</i> contains a mouse-drag value, <i>parameters\$</i> must contain <i>Coords=x1,y1,x2,y2</i>.</li> </ul> <p>See Appendix E for a list of mouse click and drag values.</p>                                                                                                                                                                                                                                                                                                                                                                                                  |
| <i>recMethod\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <i>ID=%</i>. The object's internal Windows ID.</li> <li>▶ <i>Name=\$</i>. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be <i>Command1</i>.</li> <li>▶ <i>ObjectIndex=%</i>. The number of the object among all objects of the same type in the same window.</li> <li>▶ <i>State=\$</i>. An optional qualifier for any other recognition method. There are two possible values for this setting: <i>Enabled</i> and <i>Disabled</i>. The default state is the state of the current context window (as set in the most recent <i>Window SetContext</i> command), or <i>Enabled</i> if the state has not been otherwise declared.</li> <li>▶ <i>Text=\$</i>. The text displayed on the object.</li> <li>▶ <i>VisualText=\$</i>. An optional setting used to identify an object by its visible text. It is for user clarification only and does not affect object recognition.</li> </ul> |

▶ ▶ ▶

## SpinControlVP

▶ ▶ ▶

| Syntax Element       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>parameters</i> \$ | <b>Valid values:</b> <ul style="list-style-type: none"><li>▶ <b>Coords=<i>x</i>,<i>y</i>.</b> If <i>action</i>% is a mouse click, specifies the coordinates of the click, relative to the top left of the object.</li><li>▶ <b>Coords=<i>x1</i>,<i>y1</i>,<i>x2</i>,<i>y2</i>.</b> If <i>action</i>% is a mouse drag, specifies the coordinates, where <i>x1</i>,<i>y1</i> are the starting coordinates of the drag, and <i>x2</i>,<i>y2</i> are the ending coordinates. The coordinates are relative to the top left of the object.</li></ul> |

Comments None.

Example This example clicks the first spin control in the window (ObjectIndex=1) at *x,y* coordinates of 50,25.

```
SpinControl Click, "ObjectIndex=1", "Coords=50,25"
```

See Also SpinControlVP

## SpinControlVP

Verification Point Command

▶▶SQA▶

Description Establishes a verification point for a spin control.

Syntax *Result* = **SpinControlVP** (*action*%,*recMethod*\$,*parameters*\$)

| Syntax Element  | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action</i> % | <b>The type of verification to perform. Valid values:</b> <ul style="list-style-type: none"><li>▶ <b>CompareNumeric.</b> Captures the numeric value of the text of the object and compares it to the value of <i>parameters</i>\$ Value or Range. <i>parameters</i>\$ VP and either Value or Range are required; ExpectedResult and Wait are optional.</li><li>▶ <b>CompareProperties.</b> Captures object properties information for the object and compares it to a recorded baseline. <i>parameters</i>\$ VP is required; ExpectedResult and Wait are optional.</li><li>▶ <b>CompareText.</b> Captures the text of the object and compares it to a recorded baseline. <i>parameters</i>\$ VP and Type are required; ExpectedResult and Wait are optional.</li></ul> |

▶ ▶ ▶

▶ ▶ ▶

| Syntax Element      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>recMethod\$</i>  | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <i>ID=%</i>. The object's internal Windows ID.</li> <li>▶ <i>Name=\$</i>. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be <code>Command1</code>.</li> <li>▶ <i>ObjectIndex=%</i>. The number of the object among all objects of the same type in the same window.</li> <li>▶ <i>Text=\$</i>. The text displayed on the object.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <i>parameters\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <i>ExpectedResult=%</i>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> <li>- <i>PASS</i>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail.</li> <li>- <i>FAIL</i>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail.</li> </ul> </li> <li>▶ <i>Range=&amp; , &amp;</i>. Used with the action <code>CompareNumeric</code> when a numeric range comparison is being performed, as in <i>Range=2 , 12</i> (test for numbers in this range). The values are inclusive.</li> <li>▶ <i>Type=\$</i>. Specifies the verification method to use for <code>CompareText</code> actions. The possible values are: <code>CaseSensitive</code>, <code>CaseInsensitive</code>, <code>FindSubStr</code>, <code>FindSubStrI</code> (case insensitive), and <code>UserDefined</code>. See <i>Comments</i> for more information. If <code>UserDefined</code> is specified, two additional parameters are required: <ul style="list-style-type: none"> <li>- <i>DLL=\$</i>. The full path and file name of the library that contains the function</li> <li>- <i>Function=\$</i>. The name of the custom function to use in comparing the text</li> </ul> </li> <li>▶ <i>Value=&amp;</i>. Used with the action <code>CompareNumeric</code> when a numeric equivalence comparison is being performed, as in <i>Value=25</i> (test against the value 25).</li> <li>▶ <i>VP=\$</i>. The verification point ID. IDs must be unique within a script. Required for all verification points.</li> <li>▶ <i>Wait=% , %</i>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <i>Wait=10 , 40</i> (retry the test every 10 seconds, but time out the test after 40 seconds).</li> </ul> |

## SQAConsoleClear

**Comments** This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

With the `Type=$` parameter, `CaseSensitive` and `CaseInsensitive` require a full match between the current baseline text and the text captured during playback. With `FindSubStr` and `FindSubStrI`, the current baseline can be a substring of the text captured during playback. The substring can appear anywhere in the playback text. To modify the current baseline text, double-click the verification point name in the Robot Asset pane (to the left of the script).

**Example** This example captures the properties of the first spin control in the window (`ObjectIndex=1`) and compares them to the recorded baseline in verification point `TEST1A`.

```
Result=SpinControlVP(CompareProperties,"ObjectIndex=1","VP=TEST1A")
```

**See Also** `SpinControl`

## SQAConsoleClear

Utility Command

»»SQA»

**Description** Clears the text currently displayed in the Robot console window.

**Syntax** `SQAConsoleClear`

**Comments** None.

**Example** This example clears the contents of the Robot console window.

```
SQAConsoleClear
```

**See Also** `SQAConsoleWrite`

## SQAConsoleWrite

Utility Command

»»SQA»

**Description** Writes the specified text to the Robot console window.

**Syntax** `SQAConsoleWrite text$`

| Syntax Element      | Description                                    |
|---------------------|------------------------------------------------|
| <code>text\$</code> | The text to write to the Robot console window. |

- Comments**     The message remains in the Robot console window until you clear it with `SQAConsoleClear` or until it is overwritten.
- The `SQAConsoleWrite` command includes a carriage return/line feed with the line of text.
- Use the command `Chr$(13)` to insert a carriage return into the message. For example, this command adds a blank line between “Line1” and “Line2”:
- ```
SQAConsoleWrite "Line1" + Chr$(13) + Chr$(13) + "Line2"
```
- Example** This example writes the text “Start of Playback” to the Robot console window.
- ```
SQAConsoleWrite "Start of Playback"
```
- See Also**        `SQAConsoleClear`

## SQADatapoolClose

Datapool Command



**Description**    Closes the specified datapool.

**Syntax**         `return& = SQADatapoolClose (datapool_id&)`

| Syntax Element                | Description                                                                      |
|-------------------------------|----------------------------------------------------------------------------------|
| <code>datapool_id&amp;</code> | An ID returned by <code>SQADatapoolOpen</code> specifying the datapool to close. |

**Comments**        This command requires that you include the header file `SQAUTIL.SBH` with the `SQABasic` metacommand `'$Include`.

This function has the following possible return values (Long):

|                                 |      |
|---------------------------------|------|
| <code>sqaDpSuccess</code>       | 0    |
| <code>sqaDpUninitialized</code> | -1   |
| <code>sqaDpFailure</code>       | -2   |
| <code>sqaDpExtendedError</code> | -999 |

**Example**         This example opens a datapool named `repo_dp1` and then closes it.

```
'$Include "sqautil.sbh"
DIM dp_id as Long
DIM dp_Result as Long
dp_id=SQADatapoolOpen ("repo_dp1", FALSE, SQA_DP_SEQUENTIAL, FALSE)
dp_Result = SQADatapoolClose (dp_id)
```

**See Also**        `SQADatapoolOpen`

## SQADatapoolFetch

Datapool Command



**Description** Moves the datapool cursor to the next row.

**Syntax** `return& = SQADatapoolFetch (datapool_id&)`

| Syntax Element                | Description                                                         |
|-------------------------------|---------------------------------------------------------------------|
| <code>datapool_id&amp;</code> | An ID returned by SQADatapoolOpen that represents an open datapool. |

**Comments** This command requires that you include the header file SQAUTIL.SBH with the SQABasic metacommand '\$Include.

This function has the following possible return values (Long):

|                                 |      |
|---------------------------------|------|
| <code>sqaDpSuccess</code>       | 0    |
| <code>sqaDpUninitialized</code> | -1   |
| <code>sqaDpFailure</code>       | -2   |
| <code>sqaDpEOF</code>           | -3   |
| <code>sqaDpExtendedError</code> | -999 |

SQADatapoolFetch retrieves the next row in the datapool. The "next row" in the datapool is determined by the arguments you set in SQADatapoolOpen.

If cursor wrapping is disabled, and the last row of the datapool has been retrieved, a call to SQADatapoolFetch returns `sqaDpEOF`. If SQADatapoolValue is called after `sqaDpEOF` is returned, a runtime error occurs. (Cursor wrapping is disabled when the `wrap` argument of SQADatapoolOpen is `False`.)

**Example** This example opens a datapool named `repo_dp1`, moves the cursor to the next row, and then closes the datapool.

```
'$Include "sqautil.sbh"
DIM dp_id as Long
DIM dp_Result as Long
dp_id=SQADatapoolOpen ("repo_dp1", FALSE, SQA_DP_SEQUENTIAL, FALSE)
dp_Result = SQADatapoolFetch (dp_id)
dp_Result = SQADatapoolClose (dp_id)
```

**See Also** SQADatapoolOpen SQADatapoolValue  
SQADatapoolRewind

# SQADatapoolOpen

Datapool Command

▶▶▶SQA▶

**Description** Opens the specified datapool and provides information about the datapool cursor.

**Syntax** `return& = SQADatapoolOpen ("name$", [wrap], [sequence], [exclusive])`

| Syntax Element  | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>name\$</i>   | The name of the datapool to open.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <i>wrap</i>     | <p>A optional <code>Variant</code> that indicates whether the datapool cursor should return to the first row in the row access order after the last row has been reached. Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>True</code>. After the last row in the access order has been reached, the cursor returns to the first row.</li> <li>▶ <code>False</code>. The default. After the last row in the access order has been reached, datapool access ends. If you attempt to retrieve a datapool value after the end of the datapool is reached, a runtime error occurs.</li> </ul> <p>To ensure that unique datapool rows are fetched, specify <code>False</code>, and make sure the datapool has at least as many rows as the number of users (and user iterations) that will be requesting rows at runtime.</p> <p>This argument is ignored when <i>sequence</i> is <code>SQA_DP_RANDOM</code>.</p> |
| <i>sequence</i> | <p>An optional <code>Variant</code> that determines row access order. Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>SQA_DP_SEQUENTIAL</code>. The default. Datapool access is in <b>sequential order</b>. Access begins with the first row stored in the datapool file, and it ends with the last row.</li> <li>▶ <code>SQA_DP_RANDOM</code>. Datapool access is in <b>random order</b>. Rows are retrieved in any order, and any given row can be retrieved multiple times or not at all.</li> <li>▶ <code>SQA_DP_SHUFFLE</code>. Datapool access is in <b>shuffled order</b>. Each time Robot or LoadTest rearranges, or "shuffles," the access order of all datapool rows, a unique sequence results. Each row is referenced in a shuffled sequence only once.</li> </ul>                                                                                                                              |

▶ ▶ ▶

## SQADatapoolOpen

► ► ►

| Syntax Element   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>exclusive</i> | <p>An optional <code>variant</code> that indicates whether the datapool cursor is shared with other users or is exclusive for an individual user. Valid values:</p> <ul style="list-style-type: none"> <li>► <code>True</code>. Indicates exclusive use of the datapool cursor. Each user's cursor operates independently of the others.</li> <li>► <code>False</code>. The default. Indicates a shared datapool cursor.</li> </ul> <p>This argument applies only to GUI scripts played back within a <code>LoadTest</code> schedule.</p> |

**Comments** This command requires that you include the header file `SQAUTIL.SBH` with the `SQABasic` metacommand `'$Include`.

Upon successful execution, this function returns a handle (a positive `Long` value) for subsequent datapool commands. If the command is not successful, one of these values is returned (`Long`):

```
sqaDpUninitialized -1
sqaDpFailure -2
sqaDpInvalidArgument -998
sqaDpExtendedError -999
```

With a shared cursor (`shareType=True`), all users work from the same access order. For example, if the access order for a `Colors` column is `Red`, `Blue`, and `Green`, the first user to request a value is assigned `Red`, the second is assigned `Blue`, and the third is assigned `Green`.

With a private cursor (`shareType=False`), each user starts at the top of its exclusive access order. With random order (`sequence=SQA_DP_RANDOM`) or shuffle access (`sequence=SQA_DP_SHUFFLE`), the access order is unique for each user. With sequential access (`sequence=SQA_DP_SEQUENTIAL`), the access order is the same for each user (ranging from the first row stored in the file to the last).

When using a private cursor with a sequential access order, you typically have each user run multiple iterations of the script. If each user runs a single iteration of the script, they each would access the same datapool row (the first row in the datapool).

Think of non-sequential access order (`SQA_DP_SHUFFLE` and `SQA_DP_RANDOM`) as being like a shuffled deck of cards. With `SQA_DP_SHUFFLE` access order, each time you pick a card (access a row), you remove the card from the pack. But with `SQA_DP_RANDOM` access order, the selected card is returned somewhere in the pack, making it available for selection again.



Also, with `SQA_DP_SHUFFLE`, after you reach the last card in the pack, you either reshuffle the pack and start again (`wrap=True`), or no more selections are made (`wrap=False`).

With `SQA_DP_RANDOM`, you never reach the end of the pack (there is no end-of-file condition) so `wrap` is ignored).

If multiple users (GUI users and/or virtual users) access the same datapool in a LoadTest schedule, the datapool cursor is managed as follows:

- ▶ For shared cursors, the first call to `SQADatapoolOpen` initializes the cursor. In the same schedule run, users that subsequently call `SQADatapoolOpen` to open the same datapool share the initialized cursor.
- ▶ For private cursors, the first call to `SQADatapoolOpen` initializes the user's private cursor. In the user's subsequent calls to `SQADatapoolOpen` in the same schedule, the cursor is set to the last row accessed by that user.

In `SQABasic`, `SQADatapoolOpen` is the only way to define the datapool's cursor and row access order. Unlike the VU scripting language, `SQABasic` does not include a `DATAPOOL_CONFIG` statement.

Only one emulated user can exist during Robot playback. If a script contains multiple transactions, or if a shell script executes multiple scripts, playback is considered to be one user performing multiple transactions. As a result, the concept of a shared cursor doesn't apply when scripts are played back in Robot.

## Example

This example opens a datapool named `repo_dp1` using the default access order settings. In this example, the datapool is opened for sequential access. All users share the same cursor, meaning that the first user to request a row retrieves the first row in the file, the second user retrieves the second row, and so on. After the last row in the datapool is reached, access to the datapool ends.

```
'$Include "sqautil.sbh"
DIM dp_id as Long
dp_id = SQADatapoolOpen ("repo_dp1", FALSE, SQA_DP_SEQUENTIAL, FALSE)
```

This example opens a datapool named `repo_dp2`. In this example, the datapool is opened for shuffle access. Each user maintains an exclusive cursor, meaning that each user retrieves rows according to a unique access order. After a user reaches the last row in the datapool, the user's exclusive cursor returns to the first row.

```
'$Include "sqautil.sbh"
DIM dp_id as Long
dp_id = SQADatapoolOpen ("repo_dp2", TRUE, SQA_DP_SHUFFLE, TRUE)
```

This example opens a datapool named `repo_dp3`. In this example, the datapool is opened for random access. All users share the same cursor, meaning that the first user to request a row retrieves the first row in the random order, the second user retrieves the second row, and so on. Because rows can appear in the access order

## SQADatapoolRewind

multiple times, there is no actual end to the access order. Therefore, the *wrap* argument is ignored.

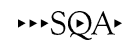
```
'$Include "sqautil.sbh"
DIM dp_id as Long
dp_id = SQADatapoolOpen ("repo_dp3", FALSE, SQA_DP_RANDOM, FALSE)
```

### See Also

SQADatapoolClose  
SQADatapoolFetch  
SQADatapoolRewind  
SQADatapoolValue

## SQADatapoolRewind

### Datapool Command



**Description** Resets the datapool cursor to the start of the datapool access order.

**Syntax** `return& = SQADatapoolRewind (datapool_id&)`

| Syntax Element                | Description                                                         |
|-------------------------------|---------------------------------------------------------------------|
| <code>datapool_id&amp;</code> | An ID returned by SQADatapoolOpen that represents an open datapool. |

**Comments** This command requires that you include the header file SQAUTIL.SBH with the SQABasic metacommand '\$Include.

This function has the following possible return values (Long):

|                                 |      |
|---------------------------------|------|
| <code>sqaDpSuccess</code>       | 0    |
| <code>sqaDpUninitialized</code> | -1   |
| <code>sqaDpFailure</code>       | -2   |
| <code>sqaDpExtendedError</code> | -999 |

This command rewinds the private cursor for the datapool referenced by `datapool_id`.

The datapool is rewound as follows:

- ▶ With datapools opened for SQA\_DP\_SEQUENTIAL access, SQADatapoolRewind resets the cursor to the first record in the datapool file.
- ▶ With datapools opened for SQA\_DP\_RANDOM or SQA\_DP\_SHUFFLE access, SQADatapoolRewind restarts the random number sequence.
- ▶ With datapools opened for SQA\_DP\_SHARED access, SQADatapoolRewind has no effect.

See the *sequence* argument of `SQADatapoolOpen` for descriptions of `SQA_DP_SEQUENTIAL`, `SQA_DP_RANDOM`, and `SQA_DP_SHUFFLE`.

At the start of a test, datapool cursors always point to the first row.

If you rewind the datapool during a test, previously accessed rows will be fetched again.

**Example** This example opens a datapool named `repo_dp1`, moves the cursor to the next row, resets the cursor, and then closes the datapool.

```
'$Include "squtil.sbh"
DIM dp_id as Long
DIM dp_Result as Long
dp_id=SQADatapoolOpen ("repo_dp1", FALSE, SQA_DP_SEQUENTIAL, FALSE)
dp_Result = SQADatapoolFetch (dp_id)
dp_Result = SQADatapoolRewind (dp_id)
dp_Result = SQADatapoolClose (dp_id)
```

**See Also** `SQADatapoolFetch`

## SQADatapoolValue

Datapool Command



**Description** Retrieves the value of the specified datapool column.

**Syntax** `return& = SQADatapoolValue (datapool_id&, column, value$)`

| Syntax Element                | Description                                                                                                                                                                                                                                                                                                                    |
|-------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>datapool_id&amp;</code> | An ID returned by <code>SQADatapoolOpen</code> that represents an open datapool.                                                                                                                                                                                                                                               |
| <code>column</code>           | A Variant that specifies the name or ID of the datapool column to retrieve. The value can be either a number (a Long or Integer) indicating the column number, or a String indicating the column name. Column names are case sensitive.<br><br>The column is in the current row retrieved with <code>SQADatapoolFetch</code> . |
| <code>value\$</code>          | Contains the value from the datapool column upon successful return.                                                                                                                                                                                                                                                            |

## SQADatapoolValue

**Comments** This command requires that you include the header file SQAUTIL.SBH with the SQABasic metacommand '\$Include.

This function has the following possible return values (Long):

|                      |      |
|----------------------|------|
| sqaDpSuccess         | 0    |
| sqaDpUninitialized   | -1   |
| sqaDpFailure         | -2   |
| sqaDpInvalidArgument | -998 |
| sqaDpExtendedError   | -999 |

If cursor wrapping is disabled, and the last row of the datapool has been retrieved, a call to SQADatapoolFetch returns sqaDpEOF. If SQADatapoolValue is called after sqaDpEOF is returned, a runtime error occurs. (Cursor wrapping is disabled when the *wrap* argument of SQADatapoolOpen is False.)

If you use a column number rather than a column name in *column*, note that the first datapool column listed in the TestManager Datapool Specification dialog box is datapool column 1.

Type checking for the *column* argument is done at runtime, since a Variant can contain data types other than a Long or String.

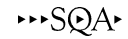
**Example** This example opens a datapool named *repo\_dp1*, moves the cursor to the next row, retrieves the value from column 1, and then closes the datapool.

```
'$Include "sqautil.sbh"
DIM dp_id as Long
DIM dp_Result as Long
dp_id = SQADatapoolOpen ("repo_dp1", FALSE, SQA_DP_SEQUENTIAL,
 FALSE)
dp_Result = SQADatapoolFetch (dp_id)
dp_Result = SQADatapoolValue (dp_id, 1, dp_Value)
dp_Result = SQADatapoolClose (dp_id)
```

**See Also** SQADatapoolFetch

# SQAEnvCreateBaseline

Utility Command



**Description** Captures a snapshot of the environment state before one or more tasks are performed that change or are suspected of changing the environment.

**Syntax** `Result = SQAEnvCreateBaseline(fileName$)`

| Syntax Element          | Description                                                                                  |
|-------------------------|----------------------------------------------------------------------------------------------|
| <code>fileName\$</code> | The name of the pre-task snapshot. This name is used as the file name for the snapshot data. |

**Comments** Returns 1 if the function call succeeds or 0 if it fails.

This command requires that you include the header file SQAUTIL.SBH with the SQABasic metacommand '\$Include.

You cannot view the snapshot data in `fileName$`. However, by calling SQAEnvCreateDelta, you can view a comparison report of two snapshot files.

You can reuse file names. Using an existing name when creating a new pre-task snapshot destroys the previous version of the snapshot.

**Example** The following code fragment captures a snapshot of the environment just before and after the application-under-test is installed.

```
'$Include "sqautil.sbh"
...

' Capture a pre-task snapshot
Result = SQAEnvCreateBaseline("PreInstall")
If Result = 0 Then
 MsgBox "Error capturing the pre-task snapshot. "
End If

' Install the application-under-test
...

' Capture a post-task snapshot
Result = SQAEnvCreateCurrent("PostInstall")
If Result = 0 Then
 MsgBox "Error capturing the post-task snapshot. "
End If
...
```

**See Also** SQAEnvCreateCurrent  
SQAEnvCreateDelta

## SQAEnvCreateCurrent

Utility Command

»»SQA»

**Description** Generates a snapshot of the environment state just after some task is performed that changes or is suspected of changing the environment.

**Syntax** *Result* = **SQAEnvCreateCurrent**(*fileName\$*)

| Syntax Element    | Description                                                                                   |
|-------------------|-----------------------------------------------------------------------------------------------|
| <i>fileName\$</i> | The name of the post-task snapshot. This name is used as the file name for the snapshot data. |

**Comments** Returns 1 if the function call succeeds or 0 if it fails.

This command requires that you include the header file SQAUTIL.SBH with the SQABasic metaccommand '\$Include.

You cannot view the snapshot data in *fileName\$*. However, by calling SQAEnvCreateDelta, you can view a comparison report of two snapshot files.

A snapshot captured with SQAEnvCreateCurrent can be used as both a post-task snapshot and, at a later point in the script, as a pre-task snapshot. For example, you might want to compare the post-task snapshot captured after you installed the application-under-test with a current snapshot taken after you perform a particular task with the application-under-test. In this case, both snapshots are created with SQAEnvCreateCurrent.

You can reuse file names. Using an existing name when creating a new post-task snapshot destroys the previous version of the snapshot.

**Example** The following code fragment captures a snapshot of the environment just before and after the application-under-test is installed.

```
'$Include "squtil.sbh"
...

' Capture a pre-task snapshot
Result = SQAEnvCreateBaseline("PreInstall")
If Result = 0 Then
 MsgBox "Error capturing the pre-task snapshot. "
End If

' Install the application-under-test
...

' Capture a post-task snapshot
Result = SQAEnvCreateCurrent("PostInstall")
If Result = 0 Then
 MsgBox "Error capturing the post-task snapshot. "
End If
```

See Also [SQAEnvCreateBaseline](#)  
[SQAEnvCreateDelta](#)

## SQAEnvCreateDelta

Utility Command

»»»SQA»

**Description** Creates a comparison report of the data captured in the pre-task and post-task snapshots.

**Syntax** `Result=SQAEnvCreateDelta (preTask$, postTask$, showReport%)`

| Syntax Element           | Description                                                                                                                                                                                                                           |
|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>preTask\$</code>   | The name of the file containing the pre-task snapshot captured with either <code>SQAEnvCreateBaseline</code> or <code>SQAEnvCreateCurrent</code> .                                                                                    |
| <code>postTask\$</code>  | The name of the file containing the post-task snapshot captured with <code>SQAEnvCreateCurrent</code> .                                                                                                                               |
| <code>showReport%</code> | Specifies whether you want to show the snapshot comparison report in a browser. Valid values: <ul style="list-style-type: none"> <li>▶ 1. Show the report in a browser.</li> <li>▶ 0. Do not show the report in a browser.</li> </ul> |

**Comments** Returns 1 if the function call succeeds or 0 if it fails.

This command requires that you include the header file `SQAUTIL.SBH` with the `SQABasic` metacommand `'$Include`.

`SQAEnvCreateDelta` compares any two snapshots of the environment. For example, you might capture and compare a snapshot of the current environment state (captured with `SQAEnvCreateCurrent`) with either of these snapshots:

- ▶ A snapshot of a “clean machine” captured with `SQAEnvCreateBaseline`. Typically, you capture a clean-machine state early in your script, before you begin to install the application-under-test or perform other tasks that might affect the environment.
- ▶ A snapshot captured previously with `SQAEnvCreateCurrent`. For example, you might want to compare the post-task snapshot captured after you installed the application-under-test with a current snapshot taken after you perform a particular task with the application-under-test. In this case, both snapshots are created with `SQAEnvCreateCurrent`.

## SQAEEnvCreateDelta

The snapshot comparison report is stored in a .HTM file. If you pass the value 1 in *showReport%*, this file is automatically displayed in a browser. If you want to locate this file yourself, you can find the path and file name as follows:

- ▶ The path is the log path plus the subdirectory \RESULTS. To find the log path, call `SQAGetLogDir`.
- ▶ The file is a .HTM file with the following root name structure:

```
postTask$ - preTask$
```

### Example

The following code fragment captures a snapshot of the environment just before and after the application-under-test is installed. It then calls `SQAEEnvCreateDelta` to compare the two snapshots and display a comparison report.

```
'$Include "sqautil.sbh"
...

' Capture a pre-task snapshot
Result = SQAEEnvCreateBaseline("PreInstall")
If Result = 0 Then
 MsgBox "Error capturing the pre-task snapshot. "
End If

' Install the application-under-test
...

' Capture a post-task snapshot
Result = SQAEEnvCreateCurrent("PostInstall")
If Result = 0 Then
 MsgBox "Error capturing the post-task snapshot. "
End If

' Compare the pre-task and post-task snapshots and
' generate a report
Result = SQAEEnvCreateDelta("PreInstall","PostInstall",1)
If Result = 0 Then
 MsgBox "Error generating the comparison report."
End If
...
```

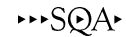
### See Also

`SQAEEnvCreateBaseline`  
`SQAEEnvCreateCurrent`  
`SQAGetLogDir`



## SQAFindObject

### Object Scripting Command



**Description** Searches for a specified object.

**Syntax** `status% = SQAFindObject(recMethod$)`

| Syntax Element           | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>recMethod\$</code> | <p>The recognition method values you use to identify an object depend on the object you're accessing. For example, if you're accessing a push button object, use the recognition method values listed for the <code>PushButton</code> user action command.</p> <p>In addition, you might need to use <code>Type=</code> to specify the object type, and/or use context notation to specify the context for the object. For details, see <i>Specifying an Object</i> in Chapter 5.</p> |

**Comments** Returns the Integer 0 (`sqaSuccess`) if `SQAFindObject` finds the specified object. If an error occurs, returns a status code that specifies the error. See the list of Object Scripting status codes in Appendix C.

This command is useful to test if an object exists before you query its properties or act upon it in some other way.

If `SQAFindObject` doesn't locate the specified object immediately, it returns `sqaObjectNotFound`. If you want to wait a certain time period for the object to appear, use `SQAWaitForObject`.

If this command acts upon a Java object, any parent Java object must be referenced in the command's `recMethod$` argument. This command ignores any parent object information in a preceding `Browser` command. For more information, see *Using Object Scripting Commands with Java Objects* in Chapter 4.

**Example** In this example, the user-defined function `CheckForOKCancelButtons()` verifies that the current window contains an OK and a Cancel button. It returns either `sqaPass` or `sqaFail`.

```
Function CheckForOKCancelButtons () As Integer
 Dim Result as Integer
 If SQAFindObject("Type=PushButton;Text=OK") = sqaSuccess And
 SQAFindObject("Type=PushButton;Text=Cancel") = sqaSuccess
 Then Result = sqaPass
 Else
 Result = sqaFail
 End If
```

## SQAGetCaptionTerminatorChar

```
'Get the recognition information for current context window
Dim CurrentWindow As Variant
SQAGetProperty "\", "Recognition", CurrentWindow
SQALogMessage Result, "Test for existence of OK and Cancel
 buttons", "Window being tested: " + CurrentWindow
CheckForOKCancelButtons = Result
End Function

'Example of using above function in a script
Sub Main
 Dim Result As Integer
 Window SetContext, "Name=myApp", ""
 MenuSelect "File->Open..."
 Result = CheckForOKCancelButtons()
End Main
```

See Also [SQAGetChildren](#)  
[SQAwaitForObject](#)

## SQAGetCaptionTerminatorChar

Utility Command

»»SQA»

**Description** Retrieves the character that Robot is currently using as the window caption terminator character.

**Syntax** `charcode% = SQAGetCaptionTerminatorChar()`

| Syntax Element         | Description                                                                    |
|------------------------|--------------------------------------------------------------------------------|
| <code>charcode%</code> | The ANSI code of the character currently being used as the caption terminator. |

**Comments** If no caption terminator is set, the return value is zero (0).

You can use `Chr$` to convert the return value into its string equivalent.

`SQAGetCaptionTerminatorChar` is the new name for the command `PLAGetCaptionTerminatorChar`.

**Example** This example retrieves the character currently being used as the Robot caption terminator, and then checks to determine if this character is the dash symbol (-).

```
charcode% = SQAGetCaptionTerminatorChar()
If Chr$(charcode%) = "-" Then
 'Caption terminator is the dash symbol...
End If
```

See Also [SQASetCaptionTerminatorChar](#)

## SQAGetChildren

### Object Scripting Command

»»SQA»

**Description** Retrieves an array containing recognition methods that identify each of an object's child objects.

**Syntax** `status% = SQAGetChildren(recMethod$, aChildren( ))`

| Syntax Element           | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>recMethod\$</code> | The recognition method values you use to identify an object depend on the object you're accessing. For example, if you're accessing a push button object, use the recognition method values listed for the <code>PushButton</code> user action command.<br><br>In addition, you might need to use <code>Type=</code> to specify the object type, and/or use context notation to specify the context for the object. For details, see <i>Specifying an Object</i> in Chapter 5. |
| <code>aChildren</code>   | An output argument that the command fills with an array of strings. This is a 0-based array that is defined as follows:<br><code>Dim aChildren() as String</code>                                                                                                                                                                                                                                                                                                              |

**Comments** Returns the Integer 0 (`sqaSuccess`) if `SQAGetChildren` executes successfully. If an error occurs, returns a status code that specifies the error. See the list of Object Scripting status codes in Appendix C.

This command requires that you include the header file `SQAUTIL.SBH` with the `SQABasic` metacommand `' $Include`.

The full recognition method for the child object is retrieved. The full recognition method includes the parent object's recognition method, a backslash (`\`), and the child object's recognition method — for example:

```
"\;Name=ParentObj;\;Name=ChildObj"
```

If this command acts upon a Java object, any parent Java object must be referenced in the command's `recMethod$` argument. This command ignores any parent object information in a preceding `Browser` command. For more information, see *Using Object Scripting Commands with Java Objects* in Chapter 4.

## SQAGetChildren

**Example** In this example, the user-defined function `TestForLabelAccelerators()` tests that all Label objects within a window have an accelerator key.

```
'$Include "squtil.sbh"
Function TestForLabelAccelerators (WindowRec As String) As Integer
 Dim Result As Integer
 Dim ChildRec() As String
 Result = sqPass
 If SQAGetChildren(WindowRec, ChildRec) = sqSuccess Then
 'cycle through children, looking for labels
 Dim ObjectType, LabelText
 Dim n As Integer
 For n = 0 To UBound(ChildRec)
 SQAGetProperty ChildRec(n), "ObjectType", ObjectType
 If ObjectType = "Label" Then
 'look for & character within each label
 SQAGetProperty ChildRec(n), "Text", LabelText
 If LabelText <> "" And InStr(LabelText, "&") Then
 SQALogMessage sqFail, "Test for label
 accelerators", "Object "" + ChildRec(n) + """"
 Result = sqFail
 End If
 End If
 Next n
 End If
 If Result = sqPass Then
 SQALogMessage sqPass, "Test for label accelerators",
 "All labels within "" + WindowRec + """"
 End If
 TestForLabelAccelerators = Result
End Function

'Example of using above function in a script
Sub Main
 Window SetContext, "Caption=Notepad - (Untitled)", ""
 MenuSelect "File->Open..."
 TestForLabelAccelerators "\;Type=Window;Caption=Open"
 Window SetContext, "Caption=Open", ""
 PushButton Click, "Text=Cancel"
End Sub
```

**See Also** [SQAFindObject](#)  
[SQAGetPropertyNames](#)  
[SQAWaitForObject](#)

# SQAGetDir

Utility Command

▶▶▶SQA▶

**Description**      Retrieves the path of standard directories used by Rational test applications.

**Syntax**            *path*\$ = **SQAGetDir**(*dirType*%)

| Syntax Element   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>dirType</i> % | <p>The type of directory to locate. Valid values:</p> <ul style="list-style-type: none"> <li>▶ SQA_DIR_PROJECT. Retrieve the path of the project directory.</li> <li>▶ SQA_DIR_REPOSITORY. Retrieve the path of the repository.</li> <li>▶ SQA_DIR_RUNFILES. Retrieve the path of the runtime files.</li> <li>▶ SQA_DIR_SCRIPTS. Retrieve the path of script files.</li> <li>▶ SQA_DIR_VPS. Retrieve the path of verification point files.</li> </ul> |

**Comments**         Returns the path of the specified directory type.

This command requires that you include the header file SQAUTIL.SBH with the SQABasic metaccommand '\$Include.

SQAGetDir retrieves paths based on the current structure of the Rational repository. If the repository structure changes in a future version of the product, some or all of the paths that SQAGetDir retrieves may not be applicable.

**Example**            This example prints to the console the paths of various standard directories.

```
'$Include "sqautil.sbh"

Sub Main

 SQAConsoleWrite "Current project: " + SQAGetDir(SQA_DIR_PROJECT)
 SQAConsoleWrite "Repository: " + SQAGetDir(SQA_DIR_REPOSITORY)
 SQAConsoleWrite "Runtime files: " + SQAGetDir(SQA_DIR_RUNFILES)
 SQAConsoleWrite "Scripts: " + SQAGetDir(SQA_DIR_SCRIPTS)
 SQAConsoleWrite "Verification points: " + SQAGetDir(SQA_DIR_VPS)

End Sub
```

**See Also**          SQAGetLogDir

SQAGetLogDir

## SQAGetLogDir

Utility Command

▶▶▶SQA▶

Description Returns the full path of the runtime log.

Syntax *logDir\$* = **SQAGetLogDir**

Comments Returns a string containing the path of the runtime log file. If the Log management GUI playback option is disabled in Robot, this command returns Robot's temporary path.

Example The following example prints the log path in the Robot console.

```
Sub Main
 Dim logPath as String
 logPath = SQAGetLogDir
 SQAConsoleWrite "The log path is " + logPath
End Sub
```

See Also **SQAEnvCreateDelta**  
**SQAGetDir**

## SQAGetOcrRegionRect

Utility Command

▶▶▶SQA▶

Description Retrieves the coordinates of the specified OCR region.

Syntax *Result*=**SQAGetOcrRegionRect**(*parameter\$*,*region%*,*rectangle*)

| Syntax Element     | Description                                                                                                                                                     |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>parameter\$</i> | Valid value:<br>VP=\$. The ID of the Window Image or Region Image verification point that tests the specified OCR region.                                       |
| <i>region%</i>     | The number of the OCR region for which you are retrieving coordinates. This number is listed in the Number column of the Image Comparator's Mask/OCR List pane. |

▶ ▶ ▶

▶ ▶ ▶

| Syntax Element   | Description                                                                                                                                                                                                                                                                             |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>rectangle</i> | <p>An output variable for the retrieved object-relative coordinates of the OCR region.</p> <p>The coordinates are returned in the following User-Defined data type:</p> <pre>Type SQARectangle   top as Integer   bottom as Integer   left as Integer   right as Integer End Type</pre> |

**Comments** Returns 1 if coordinates are successfully retrieved, or 0 if the operation fails.

The User-Defined data type *SQARectangle* is already declared and ready to use. You do not have to reference a .SBH file to use it.

This command uses the information in the baseline data file for the Window Image or Region Image verification point where the specified OCR region has been defined.

*SQAGetOcrRegionRect* must be used in conjunction with a Window Image or Region Image verification point that tests the specified OCR region.

For information about defining an OCR region, run the Image Comparator for the specified Window Image or Region Image verification point. Open the online Help and search the index for *OCR*.

**Example** The following example retrieves the coordinates of an OCR region within a window image of a Web page. The example then calculates the center of the OCR region and clicks it.

```
Sub Main
 Dim Result As Integer
 Dim ocrResult as Integer
 Dim rect as SQARectangle
 Dim coords as String

 'Initially Recorded: 1/17/00 3:38:33 PM
 'Script Name: ocr rect IE

 Result = GenericObjectVP (CompareImage, "Class=Internet
 Explorer_Server;ClassIndex=1", "VP=Window Image")

 ocrResult = SQAGetOcrRegionRect("VP=Window Image",1,rect)
```

## SQAGetOcrRegionText

```
If ocrResult = 1 Then
 'Calculate center of the OCR Region
 coords = "Coords=" + Str$(rect.left +
 (Int(rect.right-rect.left)/2)) + "," + Str$(rect.top
+
 (Int(rect.bottom-rect.top)/2))
 'Click the object at center point of OCR region
 GenericObject Click,"Class=Internet Explorer_Server;
 ClassIndex=1",coords
Else
 SQAConsoleWrite "Problem retrieving coordinates of OCR region"
End If
End Sub
```

See Also [SQAGetOcrRegionText](#)

## SQAGetOcrRegionText

Utility Command



**Description** Retrieves the text in the specified OCR region.

**Syntax** *Result* = **SQAGetOcrRegionText**(*parameter*\$, *region*%, *text*\$)

| Syntax Element      | Description                                                                                                                                                                          |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>parameter</i> \$ | Valid value: <ul style="list-style-type: none"><li>▶ VP=\$. The ID of the Window Image or Region Image verification point that tests the specified OCR region.</li></ul>             |
| <i>region</i> %     | The number of the OCR region for which you are retrieving text. This number is listed in the Number column of the Image Comparator's Mask/OCR List pane.                             |
| <i>text</i> \$      | An output variable for the retrieved text from the specified OCR region of the referenced Window Image or Region Image verification point. See <i>Comments</i> for more information. |

**Comments** Returns 1 if text is successfully retrieved, or 0 if the operation fails.

This command retrieves the text in the specified OCR region, as follows:

- ▶ Retrieves the text in the most recent Actual data file, if one exists. An Actual data file is created when the baseline data captured during recording does not match the current, or actual, data captured during playback.
- ▶ Retrieves the text in the baseline data file if no Actual data file exists.



`SQAGetOcrRegionText` must be used in conjunction with a Window Image or Region Image verification point that tests the specified OCR region.

For information about defining an OCR region, run the Image Comparator for the specified Window Image or Region Image verification point. Open the online Help and search the index for *OCR*.

### Example

The following example verifies the text in two OCR regions defined in the region image verification point B's9th.

```
Sub Main
 Dim Result as Integer
 Dim ocrResult as Integer
 Dim region as Integer
 Dim ocrText as String

 'Initially Recorded: 1/17/00 1:27:14 PM
 'Script Name: ocr text

 StartApplication ""C:\Program Files\ClassicsOnline
 \ClassicsB.exe""

 Window SetContext, "Name=frmMain", ""
 TreeView Click, "Name=treMain;\;ItemText=Beethoven",
 "Location=Button"
 TreeView Click, "Name=treMain;\;ItemText=Beethoven->Symphony
 No. 9", ""
 Result = RegionVP (CompareImage, "",
 "VP=B's9th;Coords=604,389,811,580")

 For region = 1 to 2
 ocrResult=SQAGetOcrRegionText("VP=B's9th",region,ocrText)
 If ocrResult = 1 Then
 SQAConsoleWrite "OCR region" + Str$(region) + ":"
 SQAConsoleWrite ocrText
 SQAConsoleWrite ""
 Else
 SQAConsoleWrite "Problem retrieving text for region"
 + Str$(region)
 End If
 Next region
End Sub
```

### See Also

`SQAGetOcrRegionRect`

## SQAGetProperty

### Object Scripting Command

**Description**    Retrieves the value of the specified property.

**Syntax**            `status% = SQAGetProperty(recMethod$,property$,value)`

| Syntax Element           | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>recMethod\$</code> | The recognition method values you use to identify an object depend on the object you're accessing. For example, if you're accessing a push button object, use the recognition method values listed for the <code>PushButton</code> user action command.<br><br>In addition, you might need to use <code>Type=</code> to specify the object type, and/or use context notation to specify the context for the object. For details, see <i>Specifying an Object</i> in Chapter 5. |
| <code>property\$</code>  | A case-sensitive property name. See <i>Specifying the Object Property</i> in Chapter 5 for information on the property names you can specify for a given object.                                                                                                                                                                                                                                                                                                               |
| <code>value</code>       | An output argument of type <code>Variant</code> that will contain the retrieved property value.                                                                                                                                                                                                                                                                                                                                                                                |

**Comments**        Returns the `Integer 0` (`sqaSuccess`) if `SQAGetProperty` successfully retrieves the value of the specified property. If an error occurs, returns a status code that specifies the error. See the list of Object Scripting status codes in Appendix C.

The contents of `value` is a `Variant` that's based on the native data type of the property being retrieved. For example:

- ▶ A Boolean property is retrieved as the `Integer` value 0 (for False) or -1 (for True).
- ▶ A color is retrieved as a `Long`. For example, 12632256 is retrieved for a shade of gray instead of `RGB(192,192,192)`.
- ▶ A State property for a check box is retrieved as an `Integer` (which is how the property value is stored internally). For example, if a check box is checked, the value 1 might be retrieved rather than the associated `String` value `Checked`.

To retrieve a property value in `String` form, use `SQAGetPropertyAsString`.

If the value of the specified property is stored in an array, you must specify a particular element in the array through an array index — for example:

```
Result=SQAGetProperty("Name=myList","List(0)",value)
```

Other notes about arrays of property values:

- ▶ If you don't specify an array index in a call to `SQAGetProperty`, `sqaArraysNotSupported` is returned.
- ▶ To find out how many elements are in an array, call `SQAGetPropertyArraySize`.
- ▶ To retrieve all the elements in an array, call `SQAGetPropertyArray` or `SQAGetPropertyArrayAsString`.

The maximum supported size for `Variant` strings is 32 KB. If the actual property value is larger than 32 KB, the contents of `value` is clipped to 32 KB.

If this command acts upon a Java object, any parent Java object must be referenced in the command's `recMethod$` argument. This command ignores any parent object information in a preceding `Browser` command. For more information, see *Using Object Scripting Commands with Java Objects* in Chapter 4.

#### Example

In this example, the user-defined function `CheckButton()` clicks on a check box only if it is currently unchecked.

```
Sub CheckButton (ObjectRec As String)
 Dim Result As Integer
 Dim CheckState As Variant
 'Note: A "State" of 0 means that it is unchecked
 Result = SQAGetProperty (ObjectRec, "State", CheckState)
 If Result = sqaSuccess And CheckState = 0 Then
 CheckBox Click, ObjectRec
 End If
End Sub

'Example of using above function in a script
Sub Main
 Window SetContext, "Caption=Find", ""
 CheckButton "Text=Match case"
End Sub
```

This example performs the same operation as the previous example, but without calling a user-defined function.

```
Sub Main
 Dim Result As Integer
 Dim CheckState As Variant
 Window SetContext, "Caption=Find", ""
 Result = SQAGetProperty("Type=CheckBox;Text=Match case",
 "State", CheckState)
 'Note: A "State" of 0 means that it is unchecked
 If CheckState = 0 Then
 CheckBox Click, "Text=Match case"
 End If
End Sub
```

## SQAGetPropertyArray

See Also      `SQAGetPropertyArray`                      `SQAGetPropertyNames`  
                 `SQAGetPropertyArrayAsString`              `SQASetProperty`  
                 `SQAGetPropertyArraySize`                  `SQAWaitForPropertyValue`  
                 `SQAGetPropertyAsString`

## SQAGetPropertyArray

Object Scripting Command

»»SQAS»

**Description**      Retrieves an array of values for the specified property.

**Syntax**            `status% = SQAGetPropertyArray(recMethod$,property$,`  
                         `aPropValues())`

| Syntax Element           | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>recMethod\$</code> | The recognition method values you use to identify an object depend on the object you're accessing. For example, if you're accessing a push button object, use the recognition method values listed for the <code>PushButton</code> user action command.<br><br>In addition, you might need to use <code>Type=</code> to specify the object type, and/or use context notation to specify the context for the object. For details, see <i>Specifying an Object</i> in Chapter 5. |
| <code>property\$</code>  | A case-sensitive property name. See <i>Specifying the Object Property</i> in Chapter 5 for information on the property names you can specify for a given object.                                                                                                                                                                                                                                                                                                               |
| <code>aPropValues</code> | An output argument that the command fills with an array of values for the specified property. This is a 0-based array that is defined as follows:<br><br><code>Dim aPropValues() as Variant</code>                                                                                                                                                                                                                                                                             |

**Comments**        Returns the Integer 0 (`sqaSuccess`) if `SQAGetPropertyArray` successfully retrieves the array of values for the specified property. If an error occurs, returns a status code that specifies the error. See the list of Object Scripting status codes in Appendix C.

This command requires that you include the header file `SQAUTIL.SBH` with the `SQABasic` metacommand `'$Include`.

SQAGetPropertyArray retrieves a property's value as a Variant. To retrieve the value in String form (for example, to retrieve a Boolean as "True" or "False" rather than as -1 or 0), use SQAGetPropertyArrayAsString.

If this command acts upon a Java object, any parent Java object must be referenced in the command's *recMethod\$* argument. This command ignores any parent object information in a preceding Browser command. For more information, see *Using Object Scripting Commands with Java Objects* in Chapter 4.

**Example** This example compares the contents of two list boxes.

```
'$Include "sqautil.sbh"

Sub Main
 Window SetContext, "Caption=Copy Files", ""
 Dim List1Content() As Variant
 Dim List2Content() As Variant
 'Get the contents of the two listboxes
 SQAGetPropertyArray "Type=ListBox;ObjectIndex=1", "List",
 List1Content
 SQAGetPropertyArray "Type=ListBox;ObjectIndex=2", "List",
 List2Content
 'Compare the number of elements in each listbox

 If UBound(List1Content) <> UBound(List2Content) Then
 SQALogMessage sqFail, "Dynamic Listbox Comparison",
 "Listboxes contain different number of elements"
 Else
 Dim n As Integer
 Result = sqPass
 For n = 0 to UBound(List1Content)
 If List1Content(n) <> List2Content(n) Then
 Result = sqFail
 Exit For
 End If
 Next n
 SQALogMessage Result, "Dynamic Listbox Comparison", ""
 End If

End Sub
```

See Also

|                             |                             |
|-----------------------------|-----------------------------|
| SQAGetProperty              | SQAGetPropertyNames         |
| SQAGetPropertyArrayAsString | SQAGetPropertyArraySize     |
| SQAGetPropertyArraySize     | SQAGetPropertyAsString      |
| SQAGetPropertyAsString      | SQAGetPropertyArrayAsString |
|                             | SQAGetPropertyArraySize     |
|                             | SQAGetPropertyAsString      |

## SQAGetPropertyArrayAsString

Object Scripting Command

»»SQA»

**Description** Retrieves an array of values for the specified property in `String` form.

**Syntax** `status% = SQAGetPropertyArrayAsString(recMethod$, property$, aPropValues( ))`

| Syntax Element           | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>recMethod\$</code> | The recognition method values you use to identify an object depend on the object you're accessing. For example, if you're accessing a push button object, use the recognition method values listed for the <code>PushButton</code> user action command.<br><br>In addition, you might need to use <code>Type=</code> to specify the object type, and/or use context notation to specify the context for the object. For details, see <i>Specifying an Object</i> in Chapter 5. |
| <code>property\$</code>  | A case-sensitive property name. See <i>Specifying the Object Property</i> in Chapter 5 for information on the property names you can specify for a given object.                                                                                                                                                                                                                                                                                                               |
| <code>aPropValues</code> | An output argument that the command fills with an array of values for the specified property. This is a 0-based array that is defined as follows:<br><br><code>Dim aPropValues() as String</code>                                                                                                                                                                                                                                                                              |

**Comments** Returns the `Integer 0 (sqASuccess)` if `SQAGetPropertyArrayAsString` successfully retrieves the array of values for the specified property. If an error occurs, returns a status code that specifies the error. See the list of Object Scripting status codes in Appendix C.

This command requires that you include the header file `SQAUTIL.SBH` with the `SQABasic` metacommand `'$Include`.

`SQAGetPropertyArrayAsString` retrieves values as strings. To retrieve property values in their native data types (for example, to retrieve a `Boolean` as `-1` or `0` rather than as "True" or "False"), use `SQAGetPropertyArray`.

If this command acts upon a Java object, any parent Java object must be referenced in the command's `recMethod$` argument. This command ignores any parent object information in a preceding `Browser` command. For more information, see *Using Object Scripting Commands with Java Objects* in Chapter 4.

**Example** This example logs all hidden columns within a grid.

```
'$Include "sqautil.sbh"
Sub Main
Window SetContext, "Name=myApp", ""
Dim VisibleStatus() As String
Dim n As Integer
SQAGetPropertyArrayAsString "Name=myGrid", "ColsVisible",
 VisibleStatus
'Cycle through property values, looking for hidden columns
For n = 0 to UBound(VisibleStatus)
 If VisibleStatus(n) = "False" Then
 SQALogMessage sqaNone, "Column" + Str$(n) + " is
 hidden", ""
 End If
Next n
End Sub
```

**See Also**

|                         |                         |
|-------------------------|-------------------------|
| SQAGetProperty          | SQAGetPropertyNames     |
| SQAGetPropertyArray     | SQAGetProperty          |
| SQAGetPropertyArraySize | SQAGetPropertyArraySize |
| SQAGetPropertyAsString  | SQAGetPropertyArraySize |

## SQAGetPropertyArraySize

Object Scripting Command



**Description** Retrieves the number of elements in an array of property values.

**Syntax** `status% = SQAGetPropertyArraySize(recMethod$,property$,  
size%)`

| Syntax Element     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>recMethod\$</i> | The recognition method values you use to identify an object depend on the object you're accessing. For example, if you're accessing a push button object, use the recognition method values listed for the <code>PushButton</code> user action command.<br><br>In addition, you might need to use <code>Type=</code> to specify the object type, and/or use context notation to specify the context for the object. For details, see <i>Specifying an Object</i> in Chapter 5. |
| <i>property\$</i>  | A case-sensitive property name. See <i>Specifying the Object Property</i> in Chapter 5 for information on the property names you can specify for a given object.                                                                                                                                                                                                                                                                                                               |
| <i>size%</i>       | An output value that will contain the number of elements in the array.                                                                                                                                                                                                                                                                                                                                                                                                         |

## SQAGetPropertyArraySize

**Comments** Returns the Integer 0 (*sqaSuccess*) if the property specified in *property\$* is an array. If an error occurs, returns a status code that specifies the error. See the list of Object Scripting status codes in Appendix C.

If *property\$* is not an array, but it is a valid property name, the command returns *sqaPropertyIsNotArray*

If the command returns *sqaSuccess* and *size%* is 0, the array is empty.

Calling this command is a good way to test whether a property is an array before you try to retrieve the property value with *SQAGetProperty*.

If this command acts upon a Java object, any parent Java object must be referenced in the command's *recMethod\$* argument. This command ignores any parent object information in a preceding *Browser* command. For more information, see *Using Object Scripting Commands with Java Objects* in Chapter 4.

**Example** This example logs the number of elements within a combo box.

```
Sub Main

 Dim Result As Integer
 Dim NumElements As Integer

 Window SetContext, "Name=frmMain", ""
 Result = SQAGetPropertyArraySize("Type=ComboBox;Name=Title",
 "List", NumElements)

 If Result = sqaSuccess Then
 SQALogMessage sqaNone, "The Title combobox contains" +
 Str$(NumElements) + " items", ""
 Else
 SQALogMessage sqaWarning, "Unable to obtain number of items
 in Title combobox", "Error" + Str$(Result) + ": " +
 Error$(Result)
 End If

End Sub
```

**See Also**

|                             |                         |
|-----------------------------|-------------------------|
| SQAGetProperty              | SQAGetPropertyNames     |
| SQAGetPropertyArray         | SQASetProperty          |
| SQAGetPropertyArrayAsString | SQAWaitForPropertyValue |
| SQAGetPropertyAsString      |                         |



# SQAGetPropertyAsString

Object Scripting Command

»»SQA»

**Description** Retrieves a property value in *String* form.

**Syntax** `status% = SQAGetPropertyAsString(recMethod$, property$, value)`

| Syntax Element           | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>recMethod\$</code> | The recognition method values you use to identify an object depend on the object you're accessing. For example, if you're accessing a push button object, use the recognition method values listed for the <code>PushButton</code> user action command.<br><br>In addition, you might need to use <code>Type=</code> to specify the object type, and/or use context notation to specify the context for the object. For details, see <i>Specifying an Object</i> in Chapter 5. |
| <code>property\$</code>  | A case-sensitive property name. See <i>Specifying the Object Property</i> in Chapter 5 for information on the property names you can specify for a given object.                                                                                                                                                                                                                                                                                                               |
| <code>value</code>       | An output argument of type <code>Variant</code> that will contain the retrieved property value.                                                                                                                                                                                                                                                                                                                                                                                |

**Comments** Returns the Integer 0 (`sqaSuccess`) if `SQAGetPropertyAsString` successfully retrieves the value of the specified property. If an error occurs, returns a status code that specifies the error. See the list of Object Scripting status codes in Appendix C.

`SQAGetPropertyAsString` retrieves a value as a `Variant` of `VarType 8` (`String`). Compare the command with `SQAGetProperty`, which retrieves a value as a `Variant` that's based on the native data type of the property being retrieved. For example, with `SQAGetPropertyAsString`:

- ▶ Boolean properties are retrieved as the value "True" or "False" rather than as -1 or 0.
- ▶ A color is retrieved as the string "RGB(##,##,##)", not as a `Long`. For example, a shade of gray might be retrieved as the string `RGB(192,192,192)`, not as the number 12632256.
- ▶ A State property for a check box is retrieved in *String* form. For example, if a check box is checked, the value `Checked` might be retrieved rather than the value 1 (which is how `Checked` might be stored internally).

## SQAGetPropertyAsString

If the value of the specified property is stored in an array, you must specify a particular element in the array through an array index — for example:

```
Result=SQAGetPropertyAsString("Name=myList","List(0)",value)
```

Other notes about arrays of property values:

- ▶ If you don't specify an array index in a call to `SQAGetPropertyArray`, `sqaArraysNotSupported` is returned.
- ▶ To find out how many elements are in an array, call `SQAGetPropertyArraySize`.
- ▶ To retrieve all the elements in an array, call `SQAGetPropertyArray` or `SQAGetPropertyArrayAsString`.

The maximum supported size for `Variant` strings is 32 KB. If the actual property value is larger than 32 KB, the contents of `value$` is clipped to 32 KB.

If this command acts upon a Java object, any parent Java object must be referenced in the command's `recMethod$` argument. This command ignores any parent object information in a preceding `Browser` command. For more information, see *Using Object Scripting Commands with Java Objects* in Chapter 4.

### Example

This example logs the state of the Notepad window.

```
Sub Main
 Dim Result As Integer
 Dim StateString As Variant
 Result = SQAGetPropertyAsString("\;Caption=Notepad -
 (Untitled)", "WindowState", StateString)
 If Result = sqaSuccess Then
 SQALogMessage sqaNone, "WindowState is currently: " +
 StateString, ""
 End If
End Sub
```

This example logs the background color of the TotalIncome edit box.

```
Sub Main
 Dim Result As Integer
 Dim MyColor As Variant
 Window SetContext, "Name=frmMain", ""
 Result = SQAGetPropertyAsString("Type=EditBox;
 Name=TotalIncome", "BackColor", MyColor)
 If Result = sqaSuccess Then
 SQALogMessage sqaNone, "Background color of TotalIncome: "
 + MyColor, ""
 End If
End Sub
```

### See Also

|                                          |                                      |
|------------------------------------------|--------------------------------------|
| <code>SQAGetProperty</code>              | <code>SQAGetPropertyNames</code>     |
| <code>SQAGetPropertyArray</code>         | <code>SQASetProperty</code>          |
| <code>SQAGetPropertyArrayAsString</code> | <code>SQAwaitForPropertyValue</code> |
| <code>SQAGetPropertyArraySize</code>     |                                      |

# SQAGetPropertyNames

Object Scripting Command

»»SQA»

**Description** Retrieves an array containing the names of all the object's properties.

**Syntax** `status% = SQAGetPropertyNames(recMethod$, aPropNames())`

| Syntax Element            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|---------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>recMethod\$</code>  | The recognition method values you use to identify an object depend on the object you're accessing. For example, if you're accessing a push button object, use the recognition method values listed for the <code>PushButton</code> user action command.<br><br>In addition, you might need to use <code>Type=</code> to specify the object type, and/or use context notation to specify the context for the object. For details, see <i>Specifying an Object</i> in Chapter 5. |
| <code>aPropNames()</code> | An output argument that the command fills with an array of strings. This is a 0-based array that is defined as follows:<br><br><code>Dim aPropNames() as String</code>                                                                                                                                                                                                                                                                                                         |

**Comments** Returns the Integer 0 (`sqaSuccess`) if `SQAGetPropertyNames` successfully retrieves the names of the object's properties. If an error occurs, returns a status code that specifies the error. See the list of Object Scripting status codes in Appendix C.

This command requires that you include the header file `SQAUTIL.SBH` with the `SQABasic` metacommand `'$Include`.

If this command acts upon a Java object, any parent Java object must be referenced in the command's `recMethod$` argument. This command ignores any parent object information in a preceding `Browser` command. For more information, see *Using Object Scripting Commands with Java Objects* in Chapter 4.

**Example** This example logs the total number of properties of the specified object.

```
'$Include "sqautil.sbh"
Sub Main
 Dim Result As Integer
 Dim Properties() As String
 Window SetContext, "Name=frmMain", ""
 Result = SQAGetPropertyNames("Name=cmdNext", Properties)
 If Result = sqaSuccess Then
 SQALogMessage sqaNone, "The cmdNext object has" +
 Str$(UBound(Properties)+1) + " properties", ""
 End If
End Sub
```



Syntax `longValue& = SQAGetSystemLong (code%)`

| Syntax Element              | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|-----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>longValue&amp;</code> | A system value.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <code>code%</code>          | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ SQA_MajorVersion. The major version number of Robot.</li> <li>▶ SQA_MinorVersion. The minor version number of Robot.</li> <li>▶ SQA_OS. The current operating system. Returns one of the following: <ul style="list-style-type: none"> <li>▶ SQA_OS_Win95</li> <li>▶ SQA_OS_Win2000</li> <li>▶ SQA_OS_WinNT</li> <li>▶ SQA_OS_WinNT40</li> </ul> <p>SQA_OS_Win95 represents both the Windows 95 and Windows 98 operating systems. See <i>Comments</i> for information on distinguishing between these operating systems.</p> </li> <li>▶ SQA_OS_MajorVersion. The major version number of the operating system.</li> <li>▶ SQA_OS_MinorVersion. The minor version number of the operating system.</li> </ul> |

Comments SQAGetSystemLong is the new name for the command PLAGetSystemLong.

With both Windows 95 and Windows 98 operating systems, SQA\_OS returns SQA\_OS\_WIN95, and SQA\_OS\_MajorVersion returns 4. Use SQA\_OS\_MinorVersion to distinguish between Windows 95 and Windows 98, as follows:

- ▶ With Windows 95, SQA\_OS\_MinorVersion returns 0.
- ▶ With Windows 98, SQA\_OS\_MinorVersion returns 10.

## SQAInvokeMethod

**Example** This example detects the current operating system and writes it to the Robot console.

```
Sub Main
 Dim OS As Long
 OS = SQAGetSystemLong (SQA_OS)

 Select Case OS
 Case SQA_OS_Win95
 If SQAGetSystemLong(SQA_OS_MinorVersion) = 10 Then
 SQAConsoleWrite "You are running Windows 98"
 Else
 SQAConsoleWrite "You are running Windows 95"
 End If
 Case SQA_OS_WinNT
 If SQAGetSystemLong(SQA_OS_MajorVersion) = 5 Then
 SQAConsoleWrite "You are running Windows NT 5"
 Else
 SQAConsoleWrite "You are running Windows NT 4"
 End If
 Case Else
 SQAConsoleWrite "Not sure what OS you're running"
 End Select
End Sub
```

See Also [SQASetCaptionTerminatorChar](#)

## SQAInvokeMethod

Object Scripting Command

»»SQA»

**Description** Executes the specified method of an object.

**Syntax** `status% = SQAInvokeMethod(recMethod$, objMethod$, args$)`

| Syntax Element           | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|--------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>recMethod\$</code> | The recognition method values you use to identify an object depend on the object you're accessing. For example, if you're accessing a push button object, use the recognition method values listed for the PushButton user action command.<br><br>In addition, you might need to use <code>Type=</code> to specify the object type, and/or use context notation to specify the context for the object. For details, see <i>Specifying an Object</i> in Chapter 5. |
| <code>objMethod\$</code> | The name of the method to execute.                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <code>args\$</code>      | Any arguments that the method takes. Both required and optional arguments must be specified. Separate each argument with a comma.                                                                                                                                                                                                                                                                                                                                 |

**Comments** Returns the Integer 0 (sqlSuccess) if SQLInvokeMethod successfully calls the method. If an error occurs, returns a status code that specifies the error. See the list of Object Scripting status codes in Appendix C.

SQLInvokeMethod is only supported for Visual Basic and OCX/ActiveX objects.

The only values returned are the values in the above table. This command does not return the value that the method returns, if any.

If this command acts upon a Java object, any parent Java object must be referenced in the command's *recMethod\$* argument. This command ignores any parent object information in a preceding Browser command. For more information, see *Using Object Scripting Commands with Java Objects* in Chapter 4.

**Example** This example moves the Visual Basic data control to the last record and then back to the first, to force all records to be retrieved.

```
Sub Main
 Window SetContext, "Name=frmSamples", ""
 SQLInvokeMethod "Name=datOrderInfo", "Recordset.MoveLast", ""
 SQLInvokeMethod "Name=datOrderInfo", "Recordset.MoveFirst", ""
End Sub
```

This example cycles through all customers (within a VB data control) that meet a certain criteria and then checks that they have a phone number.

```
Sub Main
 Dim Result As Integer
 Dim NoMatch As Integer
 Dim CustName, PhoneNum
 Window SetContext, "Name=frmSamples", ""
 SQLLogMessage sqlNone, "Customers starting with 'A' in their
 name...", ""
 SQLInvokeMethod "Name=datCustomerInfo", "Recordset.FindFirst",
 "CustomerName Like 'A*'"
 SQLGetProperty "Name=datCustomerInfo", "Recordset.NoMatch",
 NoMatch
 Do While Not NoMatch
 SQLGetProperty "Name=datCustomerInfo",
 "Recordset.Fields(1).Value", CustName
 SQLGetProperty "Name=datCustomerInfo",
 "Recordset.Fields(7).Value", PhoneNum
 If PhoneNum <> "" Then
 SQLLogMessage sqlPass, "Checking for phone number",
 "Customer " + CustName + " has phone number " +
 PhoneNum
 Else
 SQLLogMessage sqlFail, "Checking for phone number",
 "Customer " + CustName + " does not have a phone
 number"
 End If
 'Find next match
 SQLInvokeMethod "Name=datCustomerInfo",
 "Recordset.FindNext", "CustomerName Like 'A*'"
 SQLGetProperty "Name=datCustomerInfo",
 "Recordset.NoMatch", NoMatch
 Loop
End Sub
```

**See Also** None.

## SQALogMessage

Utility Command

»»SQA»

**Description** Writes a message to a log and optionally inserts a result flag (Pass, Fail, or Warning) in the Result column.

**Syntax** **SQALogMessage** *code%*,*message\$*,*description\$*

| Syntax Element       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>code%</i>         | Lets you insert a result flag in the Result column of the LogViewer, next to the message entry. Valid values: <ul style="list-style-type: none"> <li>▶ <i>sqaPass</i> or <i>True</i>. Inserts <i>Pass</i> in the Result column.</li> <li>▶ <i>sqaFail</i> or <i>False</i>. Inserts <i>Fail</i> in the Result column.</li> <li>▶ <i>sqaWarning</i>. Inserts <i>Warning</i> in the Result column.</li> <li>▶ <i>sqaNone</i>. Leaves the Result column blank for the message entry.</li> </ul> |
| <i>message\$</i>     | The message to insert in the log. The message appears in the Log Event column of the LogViewer.                                                                                                                                                                                                                                                                                                                                                                                             |
| <i>description\$</i> | A description of the message. The description appears in the Description field of the Log Event Properties dialog box.                                                                                                                                                                                                                                                                                                                                                                      |

**Comments** For more information about `SQALogMessage`, including an illustration of where messages are displayed, see *Displaying Messages in Robot* in Chapter 5.

To send a message to the log *and* end script execution, use `SQAScriptCmdFailure`.

`SQALogMessage` replaces the command `WriteLogMessage`.

**Example** This example writes several messages to the log.

```
Sub Main
 Dim Result As Integer
 Dim Value As Variant
 SQALogMessage sqaNone, "Starting test of Acme Application...", ""
 Window SetContext, "Caption=Acme Inc.", ""
 Result=SQAGetProperty("Type=PushButton;Text=OK","Enabled",Value)
 If Result = sqaSuccess Then
 If Value = TRUE Then
 Result = sqaPass
 Else
 Result = sqaFail
 End If
 SQALogMessage Result,"Test to see if OK button is enabled",""
 Else
 SQALogMessage sqaWarning, "Unable to perform OK button
 test", "Error" + Str$(Result) + ": " + Error$(Result)
 End If
End Sub
```



See Also [SQAScriptCmdFailure](#)

## SQAQueryKey

Utility Command

▶▶SQA▶

**Description** Returns the state of a locking key (Caps Lock, Num Lock, and Scroll Lock).

**Syntax** `state% = SQAQueryKey(keyType%)`

| Syntax Element        | Description                                                                                                                                                                                                                                                                                                    |
|-----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>keyType%</code> | One of these literal values: <ul style="list-style-type: none"> <li>▶ <code>sqaCapsLock</code>. Reports the state of the Caps Lock key.</li> <li>▶ <code>sqaNumLock</code>. Reports the state of the Num Lock key.</li> <li>▶ <code>sqaScrollLock</code>. Reports the state of the Scroll Lock key.</li> </ul> |

**Comments** Returns -1 if the key state is locked (indicator light is on), or 0 if the key state is not locked.

**Example** The following example unlocks the Num Lock feature if it is locked in the ON state.

```
If SQAQueryKey(sqaNumLock)= -1 Then
 InputKeys "{NumLock}"
End If
```

See Also [InputKeys](#)

## SQAResumeLogOutput

Utility Command

▶▶SQA▶

**Description** Resumes the output of verification point and wait state results to the log.

**Syntax** `SQAResumeLogOutput`

**Comments** `SQAResumeLogOutput` is the new name for the command `PLAResumeLogOutput`.

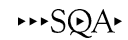
**Example** None.

SQAScriptCmdFailure

See Also [SQASuspendLogOutput](#)

## SQAScriptCmdFailure

Utility Command



**Description** Generates a script command failure and ends script execution.

**Syntax** **SQAScriptCmdFailure** *description*\$

| Syntax Element        | Description                                                                                                     |
|-----------------------|-----------------------------------------------------------------------------------------------------------------|
| <i>description</i> \$ | A description of the failure. The description appears in the Result tab of the Log Event Properties dialog box. |

**Comments** This command displays the following text in the LogViewer:

- ▶ The text “Script Command Failure” appears in the Log Event column. You can’t modify this text.
- ▶ The notation Fail appears in the Result column. You can’t modify it.
- ▶ The text you provide through this command is displayed in the Result tab of the Log Event Properties dialog box.

In addition, the description you provide of the script command failure and the line where it occurs are displayed in the Robot console window.

To send a message to the LogViewer without ending script execution, use [SQALogMessage](#).

**SQAScriptCmdFailure** is the new name for the command **PLAScriptCmdFailure**.

**Example** This example checks the current operating system and generates a script command failure if the operating system is Windows 3.x.

```
If SQAGetSystemLong (SQA_OS) = SQA_OS_Win16 then
 SQAScriptCmdFailure "This test does not work under Windows 3.x"
End If
```

See Also None.

## SQASetAssignmentChar

Utility Command



**Description** Sets the character to be used by Robot as the assignment character in SQABasic user action and verification point commands.

Syntax **SQASetAssignmentChar** *charcode%*

| Syntax Element   | Description                                                            |
|------------------|------------------------------------------------------------------------|
| <i>charcode%</i> | The ANSI code of the character to be used as the assignment character. |

Comments The `Asc` function can be used to convert the first character in a string to the correct format to be passed to the `SQASetAssignmentChar` statement.

The assignment character is used in `recMethod$` and `parameters$` arguments of SQABasic user action and verification point commands to assign values specific to that command. By default, this character is the equal sign (=).

`SQASetAssignmentChar` is the new name for the command `PLASetAssignmentChar`.

Example This example sets the Robot assignment character to be the equal sign (=).

```
SQASetAssignmentChar Asc("=")
```

See Also `SQASetSeparatorChar`

## SQASetCaptionTerminatorChar

Utility Command

»»SQA»

Description Sets the character that Robot uses as the window caption terminator character.

Syntax **SQASetCaptionTerminatorChar** *charcode%*

| Syntax Element   | Description                                                          |
|------------------|----------------------------------------------------------------------|
| <i>charcode%</i> | The ANSI code of the character to be used as the caption terminator. |

Comments The caption terminator character allows partial matches between a window caption retrieved during recording and a window caption retrieved during playback.

If this feature is enabled, Robot does not require a match of any characters that appear *after* the caption terminator character. For example, if the caption terminator is a dash ( - ), the following window captions are considered a match:

```
Mortgage Prequalifier - Customer Name
Mortgage Prequalifier - Name
```

## SQASetDefaultBrowser

To enable partial caption matching, you must select *either* of the following buttons on the Caption Matching tab of the Robot Playback Options dialog box:

- ▶ On each window search
- ▶ After automatic wait has timed out

The caption terminator character can also be set through the Robot Playback Options dialog box. A caption terminator character set through `SQASetCaptionTerminatorChar` overrides a caption terminator character set through the Playback Options dialog box.

The `Asc` function can be used to convert the first character in a string to the correct format to be passed to the `SQASetCaptionTerminatorChar` statement.

`SQASetCaptionTerminatorChar` is the new name for the command `PLASetCaptionTerminatorChar`.

**Example** This example sets the Robot caption terminator to be the dash symbol (-).

```
SQASetCaptionTerminatorChar Asc("-")
```

**See Also** `SQAGetCaptionTerminatorChar`

## SQASetDefaultBrowser

Utility Command

»»SQA»

**Description** Sets the default browser to use during playback.

**Syntax** `SQASetDefaultBrowser` (*browser*\$)

| Syntax Element    | Description                                                                                                                                                                                                                                                                                                                                                                                                                        |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>browser</i> \$ | The browser to use during playback. Valid values: <ul style="list-style-type: none"><li>▶ <code>Explorer</code>. Sets Microsoft Internet Explorer as the default browser to use during playback.</li><li>▶ <code>Navigator</code>. Sets Netscape Navigator as the default browser to use during playback.</li></ul> You can also specify a path for a particular Navigator executable. See <i>Comments</i> for syntax information. |

Comments Navigator can be set as the default browser only for playback. During recording, Internet Explorer is always used as the browser.

You can use *browser\$* to specify a path for the particular Netscape Navigator executable you want to use. For example:

```
SQASetDefaultBrowser "Navigator=c:\program files\netscape\
communicator\program\netscape.exe"
```

Using `SQASetDefaultBrowser` to change the default playback browser also changes the default browser setting as defined in the Web Browser tab of the Robot GUI Playback Options dialog box (Tools → GUI Playback Options).

If you specify an incorrect value in *browser\$*, no runtime error occurs. However, the default browser remains as defined in the Robot Web Browser tab.

If you are specifying Navigator as the default browser, and the location of Navigator's executable file isn't in the Registry, the full path must appear in the *browser\$* argument or in the Robot Web Browser tab.

To enable HTML recording, be sure that the HTML-MSIE check box is selected in the Robot Extension Manager dialog box before you begin recording against your Web page. To enable HTML playback, be sure that the HTML-MSIE and/or HTML-Navigator check boxes are selected before you play back a script against HTML objects. To display the Extension Manager dialog box in Robot, click Tools → Extension Manager.

`SQASetDefaultBrowser` can only be inserted into a script programmatically. Robot does not record this command.

Example This example sets Netscape Navigator as the default playback browser, and then uses Navigator to open Rational's Web page.

```
SQASetDefaultBrowser "Navigator"
StartBrowser "www.rational.com", "WindowTag=PlaybackNavigator"
```

See Also `StartBrowser`

## SQASetProperty

### Object Scripting Command

»»SQAS»

**Description** Assigns a value to a specified property.

**Syntax** `status% = SQASetProperty(recMethod$,property$,value)`

| Syntax Element           | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>recMethod\$</code> | The recognition method values you use to identify an object depend on the object you're accessing. For example, if you're accessing a push button object, use the recognition method values listed for the <code>PushButton</code> user action command.<br><br>In addition, you might need to use <code>Type=</code> to specify the object type, and/or use context notation to specify the context for the object. For details, see <i>Specifying an Object</i> in Chapter 5. |
| <code>property\$</code>  | A case-sensitive property name. See <i>Specifying the Object Property</i> in Chapter 5 for information on the property names you can specify for a given object.                                                                                                                                                                                                                                                                                                               |
| <code>value</code>       | A Variant containing the value you're assigning to the specified property.                                                                                                                                                                                                                                                                                                                                                                                                     |

**Comments** Returns the Integer 0 (`sqaSuccess`) if `SQASetProperty` successfully assigns `value%` to the specified property. If an error occurs, returns a status code that specifies the error. See the list of Object Scripting status codes in Appendix C.

The `value` argument can provide a value in the property's native data type (as would be retrieved by `SQAGetProperty`) or in `String` form (as would be retrieved by `SQAGetPropertyAsString`). For example, both of the following commands cause the Classics Online window to minimize:

```
Result=SQASetProperty("Caption=Classics Online","WindowState",1)
Result=SQASetProperty("Caption=Classics Online","WindowState",
 "Minimized")
```

Here are more examples of the alternatives you have for specifying `value`:

Specify Booleans as `True` or `False` (with or without quote marks) or as the Variant values `-1` or `0`.

Specify colors as a `Long`, a hexadecimal value (&H notation), or as a `String` in the form "RGB(##,##,##)". For example, a shade of gray could be specified as `12632256`, `&HC0C0C0`, or `RGB(192,192,192)`.

- ▶ Specify a State property for a check box as an `Integer` (which is how the property value is stored internally) or as a descriptive `String` value associated with each `Integer`. For example, if a check box is checked, the internal `Integer` value for State might be 1, and the associated `String` value might be `Checked`.

This command changes the value of a property for a given *instance* of an object. It does not permanently change the application under test. Closing and restarting the application undoes any change you make with `SQASetProperty`.

If the specified property has an array of values, you must specify an array index.

Many properties are not modifiable through `SQASetProperty`. For example, `SQASetProperty` can't modify `DataWindow` properties in PowerBuilder applications. If `SQASetProperty` can't modify a property, it returns the status code `sqPropertyIsReadOnly`.

You can't change a property value if it's part of an array of values.

For the following reasons, use `SQASetProperty` with caution:

- ▶ Changing a property value with `SQASetProperty` can cause unpredictable results in an application-under-test. `SQASetProperty` uses internal mechanisms for changing properties. These mechanisms may or may not trigger events within the application-under-test.
- ▶ Using `SQASetProperty` to change a property's value may not have the same effect as changing the value through some script actions. For example, the `SQASetProperty` change may not become visible until the object is redrawn.

A safer way to change a property value is to record the change as a sequence of script actions, when possible.

If this command acts upon a Java object, any parent Java object must be referenced in the command's `recMethod$` argument. This command ignores any parent object information in a preceding `Browser` command. For more information, see *Using Object Scripting Commands with Java Objects* in Chapter 4.

## Example

This example uses `SQASetProperty` to fill out certain fields of a form.

```
Sub Main
Window SetContext, "Name=frmMain", ""
'To change the value of a combobox, modify the "ItemSelected"
' property
SQASetProperty "Name=Title", "ItemSelected", "Mr."
'To change the value of an editbox, modify the "Text" property
SQASetProperty "Name=FirstName", "Text", "Michael"
SQASetProperty "Name=LastName", "Text", "Mulligan"
End Sub
```

## SQASetSeparatorChar

This example gets the data out of a specific row of a Grid OCX.

```
Sub Main
 Dim CustName, CurrentRow
 Window SetContext, "Name=frmSamples", ""
 'First get value of current row, so we can restore it later
 SQAGetProperty "Name=grdCustomer", "Row", CurrentRow
 SQASetProperty "Name=grdCustomer", "Row", 2
 SQAGetProperty "Name=grdCustomer", "Columns(1).Text", CustName
 SQALogMessage sqaNone, "Customer at row 2 of grid: " +
 CustName, ""
 'Restore row to its original setting
 SQASetProperty "Name=grdCustomer", "Row", CurrentRow
End Sub
```

See Also

|                             |                         |
|-----------------------------|-------------------------|
| SQAGetProperty              | SQAGetPropertyNames     |
| SQAGetPropertyArray         | SQAInvokeMethod         |
| SQAGetPropertyArrayAsString | SQAwaitForPropertyValue |
| SQAGetPropertyAsString      |                         |

## SQASetSeparatorChar

Utility Command



**Description** Sets the character to be used by Robot as the separator character in SQABasic commands.

**Syntax** **SQASetSeparatorChar** *charcode%*

| Syntax Element   | Description                                                           |
|------------------|-----------------------------------------------------------------------|
| <i>charcode%</i> | The ANSI code of the character to be used as the separator character. |

**Comments** The *Asc* function can be used to convert the first character in a string to the correct format to be passed to the *SQASetSeparatorChar* command.

The separator character is used to separate multiple values in SQABasic user action and verification point command arguments (*recMethod\$* and *parameters\$*). By default, this character is the semicolon (;).

*SQASetSeparatorChar* is the new name for the command *PLASetSeparatorChar*.

**Example** This example sets the Robot separator character to be the semicolon (;).

```
SQASetSeparatorChar Asc(";")
```

See Also *SQASetAssignmentChar*



# SQAShellExecute

Utility Command

»»SQAShellExecute««

**Description** Opens an application or a file.

**Syntax** **SQAShellExecute** *filename\$,directory\$,parameters\$*

| Syntax Element      | Description                                                                                                                                                             |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>filename\$</i>   | The full path and file name of the application or file.                                                                                                                 |
| <i>directory\$</i>  | The application's default directory. If you choose not to set the default directory, pass an empty string ("").                                                         |
| <i>parameters\$</i> | Optional command-line parameters to pass to the application. If <i>filename</i> specifies a non-executable file, <i>parameters</i> should contain an empty string (""). |

**Comments** If *filename* references an application, SQAShellExecute runs the application.

If *filename* references a file other than an application executable, SQAShellExecute opens the file through the application that has a Windows association with the file type. (Windows maintains associations between an application and its file types by associating the application with a particular file extension.) For example, Microsoft Word typically has a Windows association with .DOC files. If *filename* references MEMO.DOC, SQAShellExecute runs Word and opens MEMO.DOC in the Word environment.

On Windows 95 or Windows NT 4.0 platforms, *filename* can reference a Windows link file (.LNK). SQAShellExecute uses the link file to locate the application to run or the file to open. Link files are also called *shortcuts*.

On Windows 95 or Windows NT 4.0 platforms, Robot generates an SQAShellExecute command when you open an application, a file associated with an application, or a link file by clicking Start → Programs → ... or Start → Documents → ... on the Windows taskbar. Robot doesn't generate SQAShellExecute with Windows versions earlier than Windows 95 or Windows NT 4.0.

The maximum length of *filename* plus *parameters* is 259 characters.

Values you pass in *parameters* are application-specific. See the application's documentation for any supported command-line parameters.

SQAShellExecute is the new name for the command PLAShellExecute.

SQASuspendLogOutput

**Example** This example opens the Notepad text editor through the Windows 95 Start menu.

```
SQAShellExecute "C:\WIN95\Start Menu\Programs\Accessories\
Notepad.lnk", "", ""
```

**See Also** StartApplication

## SQASuspendLogOutput

Utility Command

»»SQA»

**Description** Suspends the output of verification point and wait state results to the log.

**Syntax** **SQASuspendLogOutput**

**Comments** This command is useful if you want to test for a condition without logging a result to the log.

SQASuspendLogOutput is the new name for the command  
PLASuspendLogOutput.

**Example** None.

**See Also** SQAResumeLogOutput

## SQASyncPointWait

Timing and Coordination Command

»»SQA»

**Description** Inserts a synchronization point for coordinating users in a LoadTest schedule.

**Syntax** *return&* = **SQASyncPointWait** (*syncpointID*\$)

| Syntax Element        | Description                                                                                           |
|-----------------------|-------------------------------------------------------------------------------------------------------|
| <i>syncpointID</i> \$ | A user-defined ID that identifies the synchronization point. The ID can have from 1 to 40 characters. |

**Comments** This command requires that you include the header file SQAUTIL.SBH with the SQABasic metacommand '\$Include.

SQASyncPointWait has the following possible return values (Long):

```

sqASPSuccess 0
sqASPUninitialized -1
sqASPFailure -2
sqASPExtendedError -999

```

A script pauses at a synchronization point until the release criteria specified by the schedule have been met. At that time, the script delays a random time specified in the schedule, and then resumes execution.

Typically, you will include a synchronization point in a test by inserting it into a LoadTest schedule rather than by inserting SQASyncPointWait into a script. For more information, see *Scheduling Tests* in the *Using Rational LoadTest* manual.

If you insert a synchronization point through a schedule, synchronization occurs at the beginning of the script. If you insert a synchronization point into a script through the SQASyncPointWait command, synchronization occurs at that point in the script where you inserted the command. You can insert the command anywhere in the script.

For more information about synchronization points, see the *Using Rational Robot* manual.

**Example** This example defines a sync point identified as syncpoint1.

```
SQASyncPointWait ("syncpoint1")
```

**See Also** None.

## SQAVpGetActualFileName

Utility Command



**Description** Generates a unique path and name for an actual data file used in a custom verification point.

**Syntax** `return$ = SQAVpGetActualFileName(VpName$, VpFileType$)`

| Syntax Element            | Description                                                                                                                                                                                                                            |
|---------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>VpName\$</code>     | The name of the custom verification point.                                                                                                                                                                                             |
| <code>VpFileType\$</code> | The file type of the actual data file. For example, if you are comparing data stored in .csv files, specify CSV as the file type. <code>VpFileType\$</code> can have up to three characters and must not begin with the characters VP. |

## SQAVpGetBaselineFileName

- Comments** Returns a `String` containing the full path and name for an actual data file.
- An actual data file contains the data captured for an object during the playback of a custom procedure. A baseline data file contains the data captured for the same object during a previous execution of the same custom procedure. (The actions are similar to the playback and recording of a standard verification point.)
- For the LogViewer to display the actual data, the actual data file must be in the path returned by `SQAVpGetActualFileName`.
- If the actual data captured during playback does not match the baseline data, create a file and write the actual data to it. Store this actual data file in the name and location returned by `SQAVpGetActualFileName`. These actions create an actual data file for this test and store the data file in a directory where the LogViewer expects to find it.
- For information about custom verification points, see *Managing Custom Verification Points* in Chapter 5, *Enhancements to Recorded Scripts*.

**Example** This example retrieves the full path and name of the actual data file for the custom verification point VPCHECK.

```
DIM VpActual As String
VpActual = SQAVpGetActualFileName ("VPCHECK", "CSV")
```

**See Also** `SQAVpGetBaselineFileName`  
`SQAVpLog`

## SQAVpGetBaselineFileName

Utility Command

»»SQA»

**Description** Generates a unique path and name for a baseline data file used in a custom verification point.

**Syntax** `return$ = SQAVpGetBaselineFileName(VpName$, VpFileType$)`

| Syntax Element            | Description                                                                                                                                                                                                                              |
|---------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>VpName\$</code>     | The name of the custom verification point.                                                                                                                                                                                               |
| <code>VpFileType\$</code> | The file type of the baseline data file. For example, if you are comparing data stored in .csv files, specify CSV as the file type. <code>VpFileType\$</code> can have up to three characters and must not begin with the characters VP. |

- Comments** Returns a `String` containing the full path and name for a baseline data file.
- A baseline data file contains the data captured for an object during the execution of a custom procedure. An actual data file contains the data captured for the same object during the subsequent playback of the same custom procedure. (The actions are similar to the recording and playback of a standard verification point.)
- For the LogViewer to display the baseline data, the baseline data file must be in the path returned by `SQAVpGetBaselineFileName`.
- During the playback of a test, copy the current baseline data file pointed to by `SQAVpGetCurrentBaselineFileName` to the name and location returned by `SQAVpGetBaselineFileName`. This action stores the baseline data file (called a logged baseline data file) into the directory where the LogViewer expects to find it for this test.
- For information about custom verification points, see *Managing Custom Verification Points* in Chapter 5, *Enhancements to Recorded Scripts*.

- Example** This example retrieves the full path and name of the baseline data file for the custom verification point VPBUTTON.

```
DIM VpBaseline As String
VpBaseline = SQAVpGetCurrentBaselineFileName ("VPBUTTON", "CSV")
```

- See Also** `SQAVpGetActualFileName`  
`SQAVpGetCurrentBaselineFileName`  
`SQAVpLog`

## SQAVpGetCurrentBaselineFileName

Utility Command

»»SQA»

- Description** Generates the path and name for the current baseline data file used in a custom verification point.

- Syntax** `return$ = SQAVpGetCurrentBaselineFileName (VpName$, VpFileType$)`

| Syntax Element            | Description                                                                                                                                                                                                                                      |
|---------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>VpName\$</code>     | The name of the custom verification point.                                                                                                                                                                                                       |
| <code>VpFileType\$</code> | The file type of the current baseline data file. For example, if you are comparing data stored in .csv files, specify CSV as the file type. <code>VpFileType\$</code> can have up to three characters and must not begin with the characters VP. |

## SQAVpLog

|          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Comments | <p>Returns a <code>String</code> containing the full path and name for a current baseline data file.</p> <p>A current baseline data file contains the data captured for an object during the execution of a custom procedure. This action is similar to recording a standard verification point with Robot.</p> <p>During the playback of a test, copy the current baseline data file pointed to by <code>SQAVpGetCurrentBaselineFileName</code> to the name and location returned by <code>SQAVpGetBaselineFileName</code>. This action stores the baseline data file (called a logged baseline data file) into the directory where the LogViewer expects to find it for this test.</p> <p>There is only one current baseline data file per custom verification point. However, in the LogViewer, there can be one logged baseline data file for each verification point entry.</p> <p>When you capture current baseline data and save it to the current baseline data file, the custom verification point referenced by <code>VpName\$</code> appears in the Robot Asset pane (to the right of the script). You might have to click View → Refresh to see it.</p> <p>For information about custom verification points, see <i>Managing Custom Verification Points</i> in Chapter 5, <i>Enhancements to Recorded Scripts</i>.</p> |
| Example  | <p>This example retrieves the full path and name of the current baseline data file for the custom verification point VPBUTTON.</p> <pre>DIM VpCurrBaseline As String VpCurrBaseline = SQAVpGetCurrentBaselineFileName ("VPBUTTON", "CSV")</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| See Also | <code>SQAVpGetBaselineFileName</code><br><code>SQAVpLog</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |

## SQAVpLog

Utility Command



---

|             |                                                                                                                                                         |
|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| Description | Writes a custom verification point record to a log and optionally inserts a result flag (Pass, Fail, or Warning) in the Result column of the LogViewer. |
| Syntax      | <code>SQAVpLog code%,name\$,description\$,baselineFile\$,actualFile\$</code>                                                                            |

| Syntax Element              | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|-----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>code%</code>          | Lets you insert a result flag in the Result column next to the message entry. Valid values: <ul style="list-style-type: none"> <li>▶ <code>sqaPass</code> or <code>True</code>. Inserts Pass in the Result column.</li> <li>▶ <code>sqaFail</code> or <code>False</code>. Inserts Fail in the Result column.</li> <li>▶ <code>sqaWarning</code>. Inserts Warning in the Result column.</li> <li>▶ <code>sqaNone</code>. Leaves the Result column blank for the message entry.</li> </ul> |
| <code>name\$</code>         | The name of the custom verification.                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <code>description\$</code>  | A description of the verification point.                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <code>baselineFile\$</code> | The full path and name of the baseline data file. Use <code>SQAVpGetBaselineFileName</code> to retrieve the path and name information. Use empty quotes to leave the entry blank.                                                                                                                                                                                                                                                                                                        |
| <code>actualFile\$</code>   | The full path and name of the actual data file. Use <code>SQAVpGetActualFileName</code> to retrieve the path and name information. Use empty quotes to leave the entry blank.                                                                                                                                                                                                                                                                                                            |

Comments For information about custom verification points, see *Managing Custom Verification Points* in Chapter 5, *Enhancements to Recorded Scripts*.

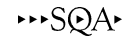
Example This example retrieves the full path and name of the baseline and actual data files for the Verification Point VPBUTTON, and then writes a log entry.

```
DIM VpBaseline As String
DIM VpActual As String
VpBaseline = SQAVpGetBaselineFileName ("VPBUTTON", "CSV")
VpActual = SQAVpGetActualFileName ("VPBUTTON", "CSV")
SQAVpLog sqaNone, "VPBUTTON", "Verify the Button Properties",
 VpBaseline, VpActual
```

See Also `SQAVpGetActualFileName`  
`SQAVpGetBaselineFileName`

## SQAWaitForObject

### Object Scripting Command



**Description** Pauses execution of the script until the specified object can be found.

**Syntax** `status% = SQAWaitForObject(recMethod$, timeout&)`

| Syntax Element            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|---------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>recMethod\$</code>  | The recognition method values you use to identify an object depend on the object you're accessing. For example, if you're accessing a push button object, use the recognition method values listed for the <code>PushButton</code> user action command.<br><br>In addition, you might need to use <code>Type=</code> to specify the object type, and/or use context notation to specify the context for the object. For details, see <i>Specifying an Object</i> in Chapter 5. |
| <code>timeout&amp;</code> | The maximum number of milliseconds to look for the object. If the object doesn't appear within the timeout period, <code>sqaTimeout</code> is returned.                                                                                                                                                                                                                                                                                                                        |

**Comments** Returns the Integer 0 (`sqaSuccess`) if `SQAWaitForObject` locates the object within the timeout period. If an error occurs, returns a status code that specifies the error. See the list of Object Scripting status codes in Appendix C.

This command is useful to call for objects that take some time to appear.

If this command acts upon a Java object, any parent Java object must be referenced in the command's `recMethod$` argument. This command ignores any parent object information in a preceding `Browser` command. For more information, see *Using Object Scripting Commands with Java Objects* in Chapter 4.

**Example** This example waits up to two minutes for a particular push button to appear.

```
Sub Main
 Dim Result As Integer
 Window SetContext, "Caption=MyApp", ""
 MenuSelect "File->Open..."
 Window SetContext, "Caption=Open", ""
 'The OK button may take a long time to appear. We can use
 ' SQAWaitForObject to synchronize our script without
 ' increasing the default wait period of all other actions
 Result = SQAWaitForObject("Type=PushButton;Text=OK", 120000)
 If Result = sqaSuccess Then
 ... 'add the rest of the actions/tests here
 End If
End Sub
```



See Also [SQFindObject](#)  
[SQGetChildren](#)

## SQAwaitForPropertyValue

Object Scripting Command

»»SQA»

**Description** Pauses execution of the script until a property is set to the specified value.

**Syntax** `status% = SQAwaitForPropertyValue(recMethod$, property$, value, timeout&)`

| Syntax Element            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|---------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>recMethod\$</code>  | The recognition method values you use to identify an object depend on the object you're accessing. For example, if you're accessing a push button object, use the recognition method values listed for the <code>PushButton</code> user action command.<br><br>In addition, you might need to use <code>Type=</code> to specify the object type, and/or use context notation to specify the context for the object. For details, see <i>Specifying an Object</i> in Chapter 5. |
| <code>property\$</code>   | A case-sensitive property name. For the command to succeed, the value of this property must match the <code>value</code> argument. See <i>Specifying the Object Property</i> in Chapter 5 for information on the property names you can specify for a given object.                                                                                                                                                                                                            |
| <code>value</code>        | A Variant that you want the specified property to be set to.                                                                                                                                                                                                                                                                                                                                                                                                                   |
| <code>timeout&amp;</code> | The maximum number of milliseconds to wait for the value of the specified property to match <code>value</code> . If the values never match within the timeout period, <code>sqaTimeout</code> is returned.                                                                                                                                                                                                                                                                     |

**Comments** Returns the Integer 0 (`sqaSuccess`) if `SQAwaitForPropertyValue` sets the specified property to `value` within the timeout period. If an error occurs, returns a status code that specifies the error. See the list of Object Scripting status codes in Appendix C.

The `value` argument can provide a value in the property's native data type (as would be retrieved by `SQGetProperty`) or in `String` form (as would be retrieved by `SQGetPropertyAsString`). For examples, see the Comments section for the `SQSetProperty` command.

## SQAwaitForPropertyValue

SQAwaitForPropertyValue waits for the object to appear, then waits for the value of the specified property to match the *value* argument. The total wait time is expressed in the *timeout* argument. For example, if *timeout* is 8000, and it takes 5 seconds for the object to appear, SQAwaitForPropertyValue will only wait an additional 3 seconds for the value of the specified property to match *value*.

If the specified property has an array of values, you must specify an array index. For example, this command specifies the fourth item in the List array:

```
Result=SQAwaitForPropertyValue("Name=ColorList","List(3)","Blue")
```

If this command acts upon a Java object, any parent Java object must be referenced in the command's *recMethod\$* argument. This command ignores any parent object information in a preceding Browser command. For more information, see *Using Object Scripting Commands with Java Objects* in Chapter 4.

### Example

This example opens a customer dialog box and waits for the OK button to be enabled before continuing.

```
Sub Main

 Dim Result As Integer

 Window SetContext, "Name=frmMain", ""
 MenuSelect "File->Open Customer..."
 Window SetContext, "Name=SelectCustomer", ""

 'Wait up to 10 seconds for the OK button to become enabled...
 Result = SQAwaitForPropertyValue("Name=cmdOK", "Enabled",
 TRUE, 10000)

 If Result <> sqaSuccess Then
 SQLogMessage sqaFail, "Waiting for cmdOK button to be
 enabled", "Error" + Str$(Result) + ": " +
 Error$(Result)
 Else
 ListBox Click, "Name=lstCustomers", "Text=Harry Houdini"
 PushButton Click, "Name=cmdOK"
 End If

End Sub
```

### See Also

|                            |                       |
|----------------------------|-----------------------|
| SQGetProperty              | SQGetPropertyAsString |
| SQGetPropertyArray         | SQGetPropertyNames    |
| SQGetPropertyArrayAsString | SQSetProperty         |
| SQGetPropertyArraySize     |                       |

## SQLClose

### Function

---

**Description** Disconnects from an ODBC data source connection that was established by `SQLOpen`.

**Syntax** `SQLClose (connection&)`

| Syntax Element               | Description                                                                      |
|------------------------------|----------------------------------------------------------------------------------|
| <code>connection&amp;</code> | A named argument that must be a long integer, returned by <code>SQLOpen</code> . |

**Comments** The return is a variant. Success returns 0 and the connection is subsequently invalid. If the connection is not valid, -1 is returned.

**Example** This example opens the data source named `SblTest`, gets the names in the ODBC data sources, and closes the connection.

```
Sub main
' Declarations
'
'
' Dim outputStr As String
' Dim connection As Long
' Dim prompt As Integer
' Dim datasources(1 To 50) As Variant
' Dim retcode As Variant
' Dim action1 as Integer
' Dim qualifier as String
'
' prompt = 5
' Open the datasource "SblTest"
' connection = SQLOpen("DSN=SblTest", outputStr, prompt:=5)
'
' action1 = 1 'Get the names of the ODBC datasources
' retcode = SQLGetSchema(connection:=connection,action:=1,
' qualifier:=qualifier, ref:=datasources())
'
' Close the datasource connection
' retcode = SQLClose(connection)
'
End Sub
```

**See Also**

|                           |                                |
|---------------------------|--------------------------------|
| <code>SQLError</code>     | <code>SQLRequest</code>        |
| <code>SQLExecQuery</code> | <code>SQLRetrieve</code>       |
| <code>SQLGetSchema</code> | <code>SQLRetrieveToFile</code> |
| <code>SQLOpen</code>      |                                |

SQLException

## SQLException

Function

---

**Description** Can be used to retrieve more detailed information about errors that might have occurred when making an ODBC function call. Returns errors for the last ODBC function and the last connection.

**Syntax** `SQLException (destination())`

| Syntax Element             | Description                                                                                                                    |
|----------------------------|--------------------------------------------------------------------------------------------------------------------------------|
| <code>destination()</code> | A two dimensional array in which each row contains one error. A named argument that is required, must be an array of variants. |

**Comments** There is no return value. The fields are: 1) character string indicating the ODBC error class/subclass, 2) numeric value indicating the data source native error code, 3) text message describing the error.

If there are no errors from a previous ODBC function call, then a 0 is returned in the caller's array at (1,1). If the array is not two dimensional or does not provide for the return of the three fields above, then an error message is returned in the caller's array at (1,1).

**Example** This example forces an error to test SQLException function.

```
sub main
' Declarations
 Dim connection As long
 Dim prompt as integer
 Dim retcode as long
 Dim errors(1 To 3, 1 To 10) as Variant
 Dim outputStr as String

' Open the datasource
connection = SQLOpen("DSN=SBLTESTW;UID=DBA;
 PWD=SQL",outputStr, prompt:=3)

' force an error to test SQLException select a
' nonexistent table
retcode = SQLExecQuery(connection:=connection,
 query:="select * from notable ")

' Retrieve the detailed error message information
' into the errors array
 SQLException destination:=errors
 retcode = SQLClose(connection)
end sub
```

**See Also** SQLClose SQLRequest  
SQLExecQuery SQLRetrieve  
SQLGetSchema SQLRetrieveToFile  
SQLOpen

## SQLExecQuery

### Function

**Description** Executes an SQL statement on a connection established by `SQLOpen`.

**Syntax** `SQLExecQuery (connection&, query$)`

| Syntax Element               | Description                                                                    |
|------------------------------|--------------------------------------------------------------------------------|
| <code>connection&amp;</code> | A named argument, required. A long integer, returned by <code>SQLOpen</code> . |
| <code>query\$</code>         | A string containing a valid SQL statement. The return is a Variant.            |

**Comments** It returns the number of columns in the result set for SQL SELECT statements; for UPDATE, INSERT, or DELETE it returns the number of rows affected by the statement. Any other SQL statement returns 0. If the function is unable to execute the query on the specified data source, or if the connection is invalid, a negative error code is returned.

If `SQLExecQuery` is called and there are any pending results on that connection, the pending results are replaced by the new results.

**Example** This example performs a query on the data source.

```
Sub main
' Declarations
'
' Dim connection As Long
' Dim destination(1 To 50, 1 To 125) As Variant
' Dim retcode As long
' Dim outputStr as String
' Dim query as String
'
' open the connection
' connection = SQLOpen("DSN=SblTest",outputStr,prompt:=3)
'
' Execute the query
' query = "select * from customer"
' retcode = SQLExecQuery(connection,query)
'
' retrieve the first 50 rows with the first 6
' columns of each row into
' the array destination, omit row numbers and put
' column names in the first row of the array
'
' retcode = SQLRetrieve(connection:=connection,
' destination:=destination, columnNames:=1,
' rowNumbers:=0,maxRows:=50,maxColumns:=6,
' fetchFirst:=0)
```

## SQLGetSchema

```
' Get the next 50 rows of from the result set
 retcode = SQLRetrieve(connection:=connection,
 destination:=destination, columnNames:=1,
 rowNumbers:=0,maxRows:=50, maxColumns:=6)
' Close the connection
 retcode = SQLClose(connection)
End Sub
```

See Also

|              |                   |
|--------------|-------------------|
| SQLClose     | SQLRequest        |
| SQLError     | SQLRetrieve       |
| SQLGetSchema | SQLRetrieveToFile |
| SQLOpen      |                   |

## SQLGetSchema

### Function

---

**Description** Returns a variety of information, including information on the data sources available, current user ID, names of tables, names and types of table columns, and other data source/database related information.

**Syntax** **SQLGetSchema** (*connection*&, *action*%, *qualifier*\$, *ref*() )

| Syntax Element      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>connection</i> & | A long integer returned by <code>SQLOpen</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <i>action</i> %     | Required. Valid values: <ol style="list-style-type: none"><li>1. List of available data sources (dimension of <code>ref()</code> is one)</li><li>2. List of databases on the current connection (not supported)</li><li>3. List of owners in a database on the current connection (not supported)</li><li>4. List of tables on the specified connection</li><li>5. List of columns in a the table specified by <i>qualifier</i>. (<code>ref()</code> must be two dimensions). Returns column name and SQL data type.</li><li>6. The user ID of the current connection user.</li><li>7. The name of the current database.</li><li>8. The name of the data source for the current connection.</li><li>9. The name of the DBMS the data source uses (for example, Oracle).</li><li>10. The server name for the data source.</li></ol> |

▶ ▶ ▶

▶ ▶ ▶

| Syntax Element     | Description                                                                                                                                                                                                                                                                         |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                    | 11. The terminology used by the data source to refer to owners.<br>12. The terminology used by the data source to refer to a table.<br>13. The terminology used by the data source to refer to a qualifier.<br>14. The terminology used by the data source to refer to a procedure. |
| <i>qualifier\$</i> | Required.                                                                                                                                                                                                                                                                           |
| <i>ref()</i>       | A Variant array for the results appropriate to the action requested. Value must be an array even if only one dimension with one element. The return is a Variant.                                                                                                                   |

**Comments** A negative return value indicates an error. A -1 is returned if the requested information cannot be found or if the connection is not valid. The destination array must be properly dimensioned to support the action or an error will be returned. Actions 2 and 3 are not currently supported. Action 4 returns all tables and does not support the use of the *qualifier*. Not all database products and ODBC drivers support all actions.

**Example** This example opens the data source named *SblTest*, gets the names in the ODBC data sources, and closes the connection.

```

Sub main
' Declarations
'
' Dim outputStr As String
' Dim connection As Long
' Dim prompt As Integer
' Dim datasources(1 To 50) As Variant
' Dim retcode As Variant
' Dim action1 as Integer
' Dim qualifier as String

prompt = 5
' Open the datasource "SblTest"
connection = SQLOpen("DSN=SblTest", outputStr, prompt:=5)

action1 = 1 'Get the names of the ODBC datasources
retcode = SQLGetSchema(connection:=connection,
 action:=1, qualifier:=qualifier, ref:=datasources())

' Close the datasource connection
retcode = SQLClose(connection)

End Sub

```

## SQLOpen

See Also      SQLClose                  SQLRequest  
                  SQLError                  SQLRetrieve  
                  SQLExecQuery            SQLRetrieveToFile  
                  SQLOpen

## SQLOpen

### Function

---

**Description**      Establishes a connection to an ODBC data source specified in *connectStr* and returns a connection ID in the return, and the completed connection string in *outputStr*. If the connection cannot be established, then a negative number ODBC error is returned.

**Syntax**            **SQLOpen** (*connectStr*\$[,*outputStr*\$][,*prompt*%])

| Syntax Element       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>connectStr</i> \$ | A named argument, a required parameter.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <i>outputStr</i> \$  | Optional.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <i>Prompt</i> %      | Optional. <i>prompt</i> specifies when the driver dialog box is displayed. Valid values: <ol style="list-style-type: none"><li>1. Driver dialog is always displayed.</li><li>2. Driver dialog is displayed only when the specification is not sufficient to make the connection.</li><li>3. The same as 2, except that dialogs that are not required are grayed and cannot be modified.</li><li>4. Driver dialog is not displayed. If the connection is not successful, an error is returned.</li></ol> When <i>prompt</i> is omitted, <b>SQLOpen</b> uses 2 as the default. |

**Comments**        The content of *connectStr* is described in the Microsoft Programmer's Reference Guide for ODBC. An example string might be "DSN=datasourcename; UID=myid; PWD=mypassword". The return must be a Long.

**Example**            This example opens the data source named *SblTest*, gets the names in the ODBC data sources, and closes the connection.

```
Sub main
 Dim outputStr, qualifier As String
 Dim connection As Long
 Dim prompt As Integer
 Dim datasources(1 To 50) As Variant
 Dim retcode As Variant
 Dim action1 as Integer
```



```

prompt = 5
' Open the datasource "SblTest"
connection = SQLOpen("DSN=SblTest", outputStr, prompt:=5)
action1 = 1 'Get the names of the ODBC datasources
retcode = SQLGetSchema(connection:=connection,
 action:=1, qualifier:=qualifier, ref:=datasources())
' Close the datasource connection
retcode = SQLClose(connection)

End Sub

```

See Also

|              |                   |
|--------------|-------------------|
| SQLClose     | SQLRequest        |
| SQLError     | SQLRetrieve       |
| SQLExecQuery | SQLRetrieveToFile |
| SQLGetSchema |                   |

## SQLRequest

### Function

**Description** Establishes a connection to the data source specified in *connectionStr*, executes the SQL statement contained in *query*, returns the results of the request in the *ref()* array, and closes the connection.

**Syntax** **SQLRequest**(*connectionStr*\$, *query*\$, *outputStr*\$, *prompt*%, *columnNames*%, *ref()*)

| Syntax Element         | Description                                                                                                                                                                                            |
|------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>connectionStr</i> § | A required argument.                                                                                                                                                                                   |
| <i>query</i> §         | A required argument.                                                                                                                                                                                   |
| <i>outputStr</i> §     | Contains the completed connection string.                                                                                                                                                              |
| <i>prompt</i> §        | An integer that specifies when driver dialog boxes are displayed (see <code>SQLOpen</code> ).                                                                                                          |
| <i>columnNames</i> §   | An integer with a value of 0 or nonzero. When <i>columnNames</i> is nonzero, column names are returned as the first row of the <i>ref()</i> array. If <i>columnNames</i> is omitted, the default is 0. |
| <i>ref()</i>           | A required argument that is a two dimensional Variant array.                                                                                                                                           |

**Comments** In the event that the connection cannot be made, the query is invalid, or other error condition, a negative number error is returned. In the event the request is successful, the positive number of results returned or rows affected is returned. Other SQL statements return 0.

The arguments are named arguments. The return is a Variant.

## SQLRetrieve

**Example** This example will open the data source SBLTESTW and execute the query specified by query and return the results in destination

```
Sub main
' Declarations
'
' Dim destination(1 To 50, 1 To 125) As Variant
' Dim prompt As integer
' Dim retcode as Variant
' Dim query as String
' Dim outputStr as String
'
' The following will open the datasource SBLTESTW and
' execute the query
' specified by query and return the results in
' destination
'
' query = "select * from class"
retcode = SQLRequest("DSN=SBLTESTW;UID=DBA;PWD=SQL",
query, outputStr, prompt, 0, destination())
End Sub
```

**See Also**

|              |                   |
|--------------|-------------------|
| SQLClose     | SQLOpen           |
| SQLError     | SQLRetrieve       |
| SQLExecQuery | SQLRetrieveToFile |
| SQLGetSchema |                   |

## SQLRetrieve

### Function

---

**Description** Fetches the results of a pending query on the connection specified by *connection* and returns the results in the *destination()* array.

**Syntax** **SQLRetrieve**(*connection*&, *destination()*, *maxColumns*%, *maxRows*%, *columnNames*%, *rowNumbers*%, *fetchFirst*%)

| Syntax Element       | Description                                                                                 |
|----------------------|---------------------------------------------------------------------------------------------|
| <i>connection</i> &  | A long.                                                                                     |
| <i>destination()</i> | A two dimensional Variant array.                                                            |
| <i>maxColumns</i> %  | An optional parameter used to specify the number of columns to be retrieved in the request. |
| <i>maxRows</i> %     | An optional parameter used to specify the number of rows to be retrieved in the request.    |
| <i>columnNames</i> % | An optional parameter that defaults to 0.                                                   |
| <i>rowNumbers</i> %  | An optional parameter that defaults to 0.                                                   |
| <i>fetchFirst</i> %  | An optional parameter that defaults to 0.                                                   |

**Comments** The return value is the number of rows in the result set or the *maxRows* requested. If the function is unable to retrieve the results on the specified connection, or if there are not results pending, -1 is returned. If no data is found, the function returns 0.

The arguments are named arguments. The return is a Variant.

If *maxColumns* or *maxRows* are omitted, the array size is used to determine the maximum number of columns and rows retrieved, and an attempt is made to return the entire result set. Extra rows can be retrieved by using `SQLRetrieve` again and by setting *fetchFirst* to 0. If *maxColumns* specifies fewer columns than are available in the result, `SQLRetrieve` discards the rightmost result columns until the results fit the specified size.

When *columnNames* is nonzero, the first row of the array will be set to the column names as specified by the database schema. When *rowNumbers* is nonzero, row numbers are returned in the first column of *destination()*. `SQLRetrieve` will clear the user's array prior to fetching the results.

When *fetchFirst* is nonzero, it causes the result set to be repositioned to the first row if the database supports the function. If the database does not support repositioning, the result set -1 error will be returned.

If there are more rows in the result set than can be contained in the *destination()* array or than have been requested using *maxRows*, the user can make repeated calls to `SQLRetrieve` until the return value is 0.

**Example** This example retrieves information from a data source.

```
Sub main
' Declarations
'
' Dim connection As Long
' Dim destination(1 To 50, 1 To 125) As Variant
' Dim retcode As long
' Dim query as String
' Dim outputStr as String
' connection = SQLOpen("DSN=SblTest",outputStr, prompt:=3)
'
' Execute the query
query = "select * from customer"
retcode = SQLExecQuery(connection,query)
'
' retrieve the first 50 rows with the first 6 columns
' of each row into the array destination, omit row
' numbers and put column names in the first row
' of the array
'
retcode = SQLRetrieve(connection:=connection,
destination:=destination, columnNames:=1,
rowNumbers:=0,maxRows:=50, maxColumns:=6,
fetchFirst:=0)
```

## SQLRetrieveToFile

```
' Get the next 50 rows of from the result set
retcode = SQLRetrieve(connection:=connection,
 destination:=destination, columnNames:=1,
 rowNumbers:=0,maxRows:=50, maxColumns:=6)

' Close the connection
retcode = SQLClose(connection)
End Sub
```

See Also

|              |                   |
|--------------|-------------------|
| SQLClose     | SQLOpen           |
| SQLError     | SQLRequest        |
| SQLExecQuery | SQLRetrieveToFile |
| SQLGetSchema |                   |

## SQLRetrieveToFile

### Function

---

**Description** Fetches the results of a pending query on the connection specified by *connection* and stores them in the file specified by *destination*.

**Syntax** **SQLRetrieveToFile**(*connection*&, *destination*\$, *columnNames*%, *columnDelimiter*\$ )

| Syntax Element            | Description                                                                                                                                                                  |
|---------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>connection</i> &       | A required argument. A long integer.                                                                                                                                         |
| <i>destination</i> \$     | A required argument. A string containing the file and path to be used for storing the results.                                                                               |
| <i>columnNames</i> %      | An integer; when nonzero, the first row of the file will be set to the column names as specified by the database schema. If <i>columnNames</i> is omitted, the default is 0. |
| <i>columnDelimiter</i> \$ | Specifies the string to be used to delimit the fields within each row. If <i>columnDelimiter</i> is omitted, a horizontal tab is used to delimit fields.                     |

**Comments** Upon successful completion of the operation, the return value is the number of rows in the result set. If the function is unable to retrieve the results on the specified connection, or if there are not results pending, -1 is returned.

The arguments are named arguments. The return is a Variant.

**Example** This example opens a connection to a data source and retrieves information to a file.

```

Sub main
' Declarations
'
' Dim connection As Long
' Dim destination(1 To 50, 1 To 125) As Variant
' Dim retcode As long
' Dim query as String
' Dim outputStr as String
' Dim filename as String
' Dim columnDelimiter as String
'
' Execute the query
'
' query = "select * from customer"
' retcode = SQLExecQuery(connection,query)
'
' Place the results of the previous query in the file
' named by filename and put the column names in the
' file as the first row. The field delimiter is %
'
' filename = "c:\myfile.txt"
' columnDelimiter = "%"
' retcode = SQLRetrieveToFile(connection:=connection,
' destination:=filename, columnNames:=1,
' columnDelimiter:=columnDelimiter)
'
' retcode = SQLClose(connection)
'
End Sub

```

**See Also**

|              |             |
|--------------|-------------|
| SQLClose     | SQLOpen     |
| SQLError     | SQLRequest  |
| SQLExecQuery | SQLRetrieve |
| SQLGetSchema |             |

## Sqr

### Function

---

**Description** Returns the square root of a number.

**Syntax** **Sqr** (*number*)

| Syntax Element | Description                                 |
|----------------|---------------------------------------------|
| <i>number</i>  | An expression containing the number to use. |

**Comments** The return value is single-precision for an Integer, Currency, or single-precision numeric expression, and double-precision for a Long, Variant, or double-precision numeric expression.

## StartApplication

**Example** This example calculates the square root of 2 as a double-precision floating point value and displays it in scientific notation.

```
Sub main
 Dim value as Double
 Dim msgtext
 value=Cdbl(Sqr(2))
 msgtext= "The square root of 2 is: " &
 Format(Value,"Scientific")
 MsgBox msgtext
End Sub
```

**See Also**

|     |     |
|-----|-----|
| Exp | Log |
| Fix | Rnd |
| Int | Sgn |

## StartApplication

Utility Command

»»SQA»

**Description** Starts the specified application from within the currently running script.

**Syntax** **StartApplication** *Pathname*\$

| Syntax Element     | Description                                                                         |
|--------------------|-------------------------------------------------------------------------------------|
| <i>Pathname</i> \$ | The full path and file name of the application to start. Arguments can be included. |

**Comments** This statement writes a message to the log indicating whether the application started successfully or failed.

To specify a quoted string within *Pathname*, use two consecutive double-quote characters (""") at the beginning and the end of the enclosed string. For example, double quotes are necessary if the file name or one of the arguments contains a space. Here are some examples:

```
StartApplication "notepad c:\autoexec.bat"
StartApplication "write "c:\program files\rational\
 rational test 7\readme.wri""
StartApplication ""c:\program files\rational\
 rational test 7\sqa7ins.exe""
```

**Example** This example starts the Windows Clock Application (and writes a message to the log indicating if the application was started successfully).

```
StartApplication "C:\WINDOWS\CLOCK.EXE"
```

**See Also** SQAShellExecute StartJavaApplication

# StartBrowser

Utility Command

»»SQA»

**Description** Starts an instance of the browser, enables Web testing, and loads a Web page if one is specified.

**Syntax** **startBrowser** [*URL*\$, ] [WindowTag=*Name*\$\_]

| Syntax Element  | Description                                                                                                                                                                                                                                   |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>URL</i> \$_  | The Universal Resource Locator of the Web page to load.                                                                                                                                                                                       |
| <i>Name</i> \$_ | An optional name that identifies this instance of the browser. In subsequent user actions, WindowTag= <i>Name</i> \$_ is used in the <i>recMethod</i> \$_ argument of the Window SetContext command to identify this instance of the browser. |

**Comments** The StartBrowser command enables Web object recognition. If you start a Web browser outside of Robot (that is, without using the StartBrowser command), you must open rbtstart.htm in your browser, or run the Rational ActiveX Test Control that rbtstart.htm references, before loading Web pages for testing. By default, rbtstart.htm is located in:

```
C:\Program Files\Rational\Rational Test 7
```

Once you run StartBrowser or the Rational ActiveX Test Control for a particular browser, Web object recognition is enabled for all subsequent actions against that browser and any new browser windows opened from that browser. For example, if you run StartBrowser to open Browser1, and then from Browser1 you open Browser2 through a JavaScript command or by holding down the Shift key and clicking on a link in Internet Explorer, Web testing is enabled for both Browser1 and Browser2.

**Example** This example enables Web object recognition and starts an instance of the Web browser (identified as Instance1). It then loads the www.rational.com Web page, which is now ready for testing.

```
startBrowser "http://www.rational.com/", "WindowTag=Instance1"
```

**See Also**

|                |             |                      |
|----------------|-------------|----------------------|
| HTMLDocument   | HTMLImageVP | HTMLTable            |
| HTMLDocumentVP | HTMLLink    | HTMLTableVP          |
| HTMLImage      | HTMLLinkVP  | SQASetDefaultBrowser |

# StartJavaApplication

Utility Command

»»SQAS»

**Description** Starts the specified Java application from within the currently running script.

**Syntax** **StartJavaApplication** **Class:=classname\$** [**CP:=classpath\$**] [**Working:=workingfolder\$**] [**JvmKey:=jvmkey\$**] [**JvmFile:=jvmfile\$**] [**JvmOpts:=jvmoptions\$**]

| Syntax Element                        | Description                                                                                                                                                                                                                                                                                                            |
|---------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>Class:=classname\$</code>       | The application's main class name.<br>The class name is required. Optionally, you can include class arguments with the class name.                                                                                                                                                                                     |
| <code>CP:=classpath\$</code>          | The class path for the class file and any required components. This value overrides any <code>classpath</code> environment setting.<br>You can specify the current directory by inserting a dot (.) into the class path.<br>If you specify multiple directories in the class path, separate them with semi-colons (;). |
| <code>Working:=workingfolder\$</code> | The full path to a working directory. Use this argument to set the working directory before running the Java application. Doing so sets the current directory to the specified working directory for that instance of the application.                                                                                 |
| <code>JvmKey:=jvmkey\$</code>         | A Robot-defined keyword that identifies a particular JVM. If a <code>JvmFile</code> is also specified, <code>JvmKey</code> takes precedence.<br>See <i>Comments</i> for the list of possible keywords.                                                                                                                 |
| <code>JvmFile:=jvmfile\$</code>       | The executable file name of a JVM. This argument can include the path as well as the file name.<br>If a <code>JvmKey</code> is also specified, <code>JvmKey</code> takes precedence.                                                                                                                                   |
| <code>JvmOpts:=jvmoptions\$</code>    | Option parameters for the specified JVM.                                                                                                                                                                                                                                                                               |

**Comments** `StartJavaApplication` uses named arguments. Named arguments can appear in any order after the command name. Further, optional named arguments can be omitted. With `StartJavaApplication`, only `Class:=classname$` is required. All other arguments are optional. For more information about named arguments, see *Passing Named Arguments* in Chapter 3.



Use `StartJavaApplication` if you want to play back the script under `Quantify` or `PureCoverage`.

If you use `StartJavaApplication`, you cannot start the Java application from a batch file.

To specify a particular JVM, you can use either `JvmKey` or `JvmFile`. Note that:

- ▶ If you specify both `JvmKey` and `JvmFile`, `JvmKey` takes precedence.
- ▶ If you specify neither `JvmKey` nor `JvmFile`, Robot uses the JavaSoft Sun JVM (`java.exe`) that it finds using the `PATH` environment variable.

`JvmKey` can take the following keyword values:

| JvmKey keyword          | JVM used                                                                                                                                                                               |
|-------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Java                    | The JavaSoft Sun JVM ( <code>java.exe</code> ) found using the <code>PATH</code> environment variable. This keyword is the default.                                                    |
| Jview                   | The Microsoft JVM ( <code>jview.exe</code> ) found using the <code>PATH</code> environment variable.                                                                                   |
| JavaSoft JDK <i>n.n</i> | The JavaSoft installed JDK JVM. Robot uses the Registry to find the executable for the JVM.<br><br>Note that <i>n.n</i> identifies the version number (for example, JavaSoft JDK 1.2). |
| JavaSoft JRE <i>n.n</i> | The JavaSoft installed JRE JVM. Robot uses the Registry to find the executable for the JVM.<br><br>Note that <i>n.n</i> identifies the version number (for example, JavaSoft JRE 1.2). |

### Example

This example runs the Java application with the class name `Notepad`. The class path is the current directory, as indicated by the dot (`.`) in the `CP` argument. Also, a particular working directory is specified on the D drive. In this example, Robot runs the JDK JVM executable file that it finds in the Registry.

```
StartJavaApplication Class:="Notepad",CP:=".",
Working:="d:\jdk1.2\demo\jfc\notepad",JvmKey:="JavaSoft JDK 1.2"
```

This example runs the Java application with the class name `AwtSimple`. Two directories are specified in the class path. In this example, Robot runs the Microsoft JVM (`jview.exe`), which it finds using the `PATH` environment variable.

```
StartJavaApplication Class:="AwtSimple",
CP:="E:\VisualCafePDE\BIN\COMPONENTS\SYMBEANS.JAR;
C:\JviewJavaApps\AwtSimple.Jar",JvmKey:="Jview"
```

StartSaveWindowPositions

See Also      None.

## StartSaveWindowPositions

Utility Command

»»SQA»

---

|             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Description | Marks the start of the script commands that save the window positions for restoration at playback.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Syntax      | <b>startSaveWindowPositions</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| Comments    | <p>When you record a script, Robot optionally saves the positions of all windows at the beginning of the recording. Scripts have <code>Window SetPosition</code> statements between <code>StartSaveWindowPositions</code> and <code>EndSaveWindowPositions</code> statements, identifying the locations and status of the windows to be restored.</p> <p><code>StartSaveWindowPositions</code> sets all playback synchronization and timeout values to zero to speed up the processing of the <code>Window</code> commands.</p> <p><code>EndSaveWindowPositions</code> resets all sync and timeout values to their default values.</p> <p>Script commands between <code>StartSaveWindowPositions</code> and <code>EndSaveWindowPositions</code> generate a <b>Warning</b> in the log if not executed properly on playback.</p> <p>If you do not want to store the window position information, you can turn off this feature in the Recording Options dialog box.</p> <p>On playback, the Unexpected Active Window checking is turned off between the <code>StartSaveWindowPositions</code> and <code>EndSaveWindowPositions</code> statements.</p> |
| Example     | <p>This example marks the start of the script commands that save the window positions for restoration at playback.</p> <pre>startSaveWindowPositions   Window SetPosition, "Caption=Text.Doc",     "Coords=455,186,161,101;Status=NORMAL"   Window MoveTo, "Caption=QuarterByte Savings Bank",     "Coords=151,10,490,248;Status=NORMAL" EndSaveWindowPositions</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| See Also    | <code>EndSaveWindowPositions</code><br><code>Window (Action - SetPosition)</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |

## StartTimer

Utility Command

»»SQA»

**Description** Starts the specified timer in the currently running GUI script and writes a message to the log.

**Syntax** `startTimer TimerID$`

| Syntax Element   | Description                    |
|------------------|--------------------------------|
| <i>TimerID\$</i> | ID of the timer to be started. |

**Comments** When the timer starts, a log message indicating when the timer was started is written to the log.

A GUI script can have up to 20 simultaneously active timers.

If the specified timer name is already in use in the GUI script, the timer is stopped and the elapsed time is reported to Rational LoadTest. The timer is then restarted, beginning with an elapsed time of 0.

**Example** This example starts timer 001, establishes a verification point for a window, and then stops timer 001.

```
startTimer "001"
Result = WindowVP (CompareMenu,
"Caption=Untitled - Notepad", "VP=QBMPSTSTA")
StopTimer "001"
```

**See Also** StopTimer

## Static

Statement

**Description** Declares variables and allocate storage space.

**Syntax** `static variableName [As type] [,variableName [As type]]`  
...

| Syntax Element      | Description                          |
|---------------------|--------------------------------------|
| <i>variableName</i> | The name of the variable to declare. |
| <i>Type</i>         | The data type of the variable.       |

Static

Comments     Variables declared with the `Static` statement retain their value as long as the program is running. The syntax of `Static` is exactly the same as the syntax of the `Dim` statement.

All variables of a procedure can be made static by using the `Static` keyword in a definition of that procedure. See `Function` or `Sub` for more information.

Example       This example puts account numbers to a file using the variable `grecord` and then prints them again.

```
Type acctrecord
 acctno as Integer
End Type

Sub main
 Static grecord as acctrecord
 Dim x
 Dim total
 Dim msgtext as String
 On Error Resume Next
 Open "C:\TEMP001" For Output as #1
 Do While grecord.acctno<>0
i: grecord.acctno=InputBox("Enter 0 or new account #" & x & ":")
 If Err<>0 then
 MsgBox "Error occurred. Try again."
 Err=0
 Goto i
 End If
 If grecord.acctno<>0 then
 Print #1, grecord.acctno
 x=x+1
 End If
 Loop
 Close #1
 total=x-1
 msgtext="The account numbers are: " & Chr(10)
 Open "C:\TEMP001" For Input as #1
 For x=1 to total
 Input #1, grecord.acctno
 msgtext=msgtext & Chr(10) & grecord.acctno
 Next x
 MsgBox msgtext
 Close #1
 Kill "C:\TEMP001"
End Sub
```

See Also     `Dim`                    `Option Base`  
              `Function`                `ReDim`  
              `End Function`        `Sub...End Sub`  
              `Global`

## StaticComboBox

### Statement

**Description** Creates a combination of a list of choices and a text box.

**Syntax** Syntax A `StaticComboBox x, y, dx, dy, text$, .field`

Syntax B `StaticComboBox x, y, dx, dy, stringarray$( ), .field`

| Syntax Element       | Description                                                                                                             |
|----------------------|-------------------------------------------------------------------------------------------------------------------------|
| <i>x, y</i>          | The upper left corner coordinates of the list box, relative to the upper left corner of the dialog box.                 |
| <i>dx, dy</i>        | The width and height of the combo box in which the user enters or selects text.                                         |
| <i>text\$</i>        | A string containing the selections for the combo box.                                                                   |
| <i>stringarray\$</i> | An array of dynamic strings for the selections in the combo box.                                                        |
| <i>.field</i>        | The name of the dialog-record field that will hold the text string entered in the text box or chosen from the list box. |

**Comments** The `StaticComboBox` statement is equivalent to the `ComboBox` or `DropComboBox` statement, but the list box of `StaticComboBox` always stays visible. All dialog functions and statements that apply to the `ComboBox` apply to the `StaticComboBox` as well.

The *x* argument is measured in 1/4 system-font character-width units. The *y* argument is measured in 1/8 system-font character-width units. (See `Begin Dialog` for more information.)

The *text\$* argument must be defined, using a `Dim` statement, before the `Begin Dialog` statement is executed. The arguments in the *text\$* string are entered as shown in the following example:

```
dimname="listchoice"+Chr$(9)+"listchoice"+Chr$(9)+"listchoice"...
```

The string in the text box will be recorded in the field designated by the *.field* argument when the OK button (or any `PushButton` other than `Cancel`) is pushed. The *field* argument is also used by the dialog statements that act on this control.

Use the `StaticComboBox` statement only between a `Begin Dialog` and an `End Dialog` statement.

## StatusBar

**Example** This example defines a dialog box with a static combo box labeled Installed Drivers and the OK and Cancel buttons.

```
Sub main
 Dim cchoices as String
 cchoices="MIDI Mapper"+Chr$(9)+"Timer"
 Begin Dialog UserDialog 182, 116, "SQABasic Dialog Box"
 StaticComboBox 7, 20, 87, 49, cchoices, .StaticComboBox1
 Text 6, 3, 83, 10, "Installed Drivers", .Text1
 OKButton 118, 12, 54, 14
 CancelButton 118, 34, 54, 14
 End Dialog
 Dim mydialogbox As UserDialog
 Dialog mydialogbox
 If Err=102 then
 MsgBox "You pressed Cancel."
 Else
 MsgBox "You pressed OK."
 End If
End Sub
```

## See Also

|              |              |                |
|--------------|--------------|----------------|
| Begin Dialog | ComboBox     | OptionGroup    |
| End Dialog   | Dialog       | Picture        |
| Button       | DropComboBox | StaticComboBox |
| ButtonGroup  | GroupBox     | Text           |
| CancelButton | ListBox      | TextBox        |
| Caption      | OKButton     |                |
| CheckBox     | OptionButton |                |

## StatusBar

### User Action Command



**Description** Performs an action on a status bar control.

**Syntax** **StatusBar** *action%*, *recMethod\$*, *parameters\$*

| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i> | One of these mouse actions: <ul style="list-style-type: none"><li>▶ <i>MouseClicked</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> must contain <i>Coords=x,y</i>.</li><li>▶ <i>MouseDown</i>. The dragging of the mouse while mouse buttons and/or shifting keys (Ctrl, Alt, Shift) are pressed. When <i>action%</i> contains a mouse-drag value, <i>parameters\$</i> must contain <i>Coords=x1,y1,x2,y2</i>.</li></ul> See Appendix E for a list of mouse click and drag values. |





| Syntax Element       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>recMethod</i> \$  | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ ID=% . The object's internal Windows ID.</li> <li>▶ Name=\$ . A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be Command1.</li> <li>▶ ObjectIndex=% . The number of the object among all objects of the same type in the same window.</li> <li>▶ State=\$ . An optional qualifier for any other recognition method. There are two possible values for this setting: Enabled and Disabled. The default state is the state of the current context window (as set in the most recent Window SetContext command), or Enabled if the state has not been otherwise declared.</li> <li>▶ VisualText=\$ . An optional setting used to identify an object by its visible text. It is for user clarification only and does not affect object recognition.</li> </ul> |
| <i>parameters</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ Coords=<i>x,y</i> . If <i>action%</i> is a mouse click, specifies the coordinates of the click, relative to the top left of the object.</li> <li>▶ Coords=<i>x1,y1,x2,y2</i> . If <i>action%</i> is a mouse drag, specifies the coordinates, where <i>x1,y1</i> are the starting coordinates of the drag, and <i>x2,y2</i> are the ending coordinates. The coordinates are relative to the top left of the object.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                   |

Comments None.

Example This example clicks the first status bar control in the window (ObjectIndex=1) at *x,y* coordinates of 50,25.

```
StatusBar Click, "ObjectIndex=1", "Coords=50,25"
```

See Also StatusBarVP

## StatusBarVP

Verification Point Command

**Description** Establishes a verification point for a status bar control.

**Syntax** `Result = StatusBarVP (action%, recMethod$, parameters$)`

| Syntax Element           | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>action%</code>     | <p>The type of verification to perform. Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>CompareData</code>. Captures the data of the object and compares it to a recorded baseline. <code>parameters\$ VP</code> and <code>Type</code> are required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li> <li>▶ <code>CompareNumeric</code>. Captures the numeric value of the text of the object and compares it to the value of <code>parameters\$ Value or Range</code>. <code>parameters\$ VP</code> and either <code>Value or Range</code> are required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li> <li>▶ <code>CompareProperties</code>. Captures object properties information for the object and compares it to a recorded baseline. <code>parameters\$ VP</code> is required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li> <li>▶ <code>CompareText</code>. Captures the text of the object and compares it to a recorded baseline. <code>parameters\$ VP</code> and <code>Type</code> are required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li> </ul> |
| <code>recMethod\$</code> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>ID=%</code>. The object's internal Windows ID.</li> <li>▶ <code>Name=\$</code>. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be <code>Command1</code>.</li> <li>▶ <code>ObjectIndex=%</code>. The number of the object among all objects of the same type in the same window.</li> </ul> <p style="text-align: right;">▶ ▶ ▶</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |





| Syntax Element       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>parameters</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <b>ExpectedResult=%</b>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> <li>- <b>PASS</b>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail.</li> <li>- <b>FAIL</b>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail.</li> </ul> </li> <li>▶ <b>Range=&amp; , &amp;</b>. Used with the action <code>CompareNumeric</code> when a numeric range comparison is being performed, as in <code>Range=2 , 12</code> (test for numbers in this range). The values are inclusive.</li> <li>▶ <b>Type=\$</b> . Specifies the verification method to use for <code>CompareText</code> actions. The possible values are: <code>CaseSensitive</code>, <code>CaseInsensitive</code> and <code>UserDefined</code>. If <code>UserDefined</code> is specified, two additional parameters are required: <ul style="list-style-type: none"> <li>- <b>DLL=\$</b>. The full path and file name of the library that contains the function</li> <li>- <b>Function=\$</b>. The name of the custom function to use in comparing the text</li> </ul> </li> <li>▶ <b>Value=&amp;</b>. Used with the action <code>CompareNumeric</code> when a numeric equivalence comparison is being performed, as in <code>Value=25</code> (test against the value 25).</li> <li>▶ <b>VP=\$</b>. The verification point ID. IDs must be unique within a script. Required for all verification points.</li> <li>▶ <b>Wait=% , %</b>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10 , 40</code> (retry the test every 10 seconds, but time out the test after 40 seconds).</li> </ul> |

**Comments** This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

**Example** This example captures the properties of the first status bar control in the window (`ObjectIndex=1`) and compares them to the recorded baseline in verification point TEST1A.

```
Result = StatusBarVP(CompareProperties,"ObjectIndex=1","VP=TEST1A")
```

**See Also** `StatusBar`

Stop

## Stop Statement

---

|             |                                                                                                                                                                                                                                 |
|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Description | Halts program execution.                                                                                                                                                                                                        |
| Syntax      | <b>stop</b>                                                                                                                                                                                                                     |
| Comments    | <code>stop</code> statements can be placed anywhere in a program to suspend its execution. Although the <code>stop</code> statement halts program execution, it does not close files or clear variables.                        |
| Example     | This example stops program execution at the user's request.<br><pre>Sub main   Dim str1   str1=InputBox("Stop program execution? (Y/N):")   If str1="Y" or str1="y" then <b>stop</b>   MsgBox "Program complete." End Sub</pre> |
| See Also    | None.                                                                                                                                                                                                                           |

## StopTimer Utility Command

---

»»SQAS»

| Description       | Stops the specified timer in the currently running GUI script and writes the elapsed time in milliseconds to the log.                                                                                                                         |                |             |                   |                                |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|-------------|-------------------|--------------------------------|
| Syntax            | <b>stopTimer</b> <i>TimerID</i> \$<br><table><thead><tr><th>Syntax Element</th><th>Description</th></tr></thead><tbody><tr><td><i>TimerID</i>\$</td><td>ID of the timer to be stopped.</td></tr></tbody></table>                              | Syntax Element | Description | <i>TimerID</i> \$ | ID of the timer to be stopped. |
| Syntax Element    | Description                                                                                                                                                                                                                                   |                |             |                   |                                |
| <i>TimerID</i> \$ | ID of the timer to be stopped.                                                                                                                                                                                                                |                |             |                   |                                |
| Comments          | When the timer stops, a log message indicating when the timer was stopped and the elapsed time in milliseconds is written to the log.                                                                                                         |                |             |                   |                                |
| Example           | This example starts timer 001, establishes a verification point for a window, and then stops timer 001.<br><pre>StartTimer "001" Result = WindowVP (CompareMenu, "Caption=Untitled - Notepad",   "VP=QBMPSTSTA") <b>stopTimer</b> "001"</pre> |                |             |                   |                                |
| See Also          | <code>startTimer</code>                                                                                                                                                                                                                       |                |             |                   |                                |

## Str

### Function

---

**Description** Returns a string representation of a number.

**Syntax** `Str[ $ ] ( number )`

| Syntax Element | Description                                                                                                                                                      |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| \$             | Optional. If specified the return type is <code>String</code> . If omitted, the function will return a <code>Variant</code> of <code>VarType 8 (String)</code> . |
| <i>number</i>  | The number to represent as a string.                                                                                                                             |

**Comments** The precision in the returned string is single-precision for an integer or single-precision numeric expression, double precision for a long or double-precision numeric expression, and currency precision for currency. Variants return the precision of their underlying `VarType`.

**Example** This example prompts for two numbers, adds them, then shows them as a concatenated string.

```
Sub main
 Dim x as Integer
 Dim y as Integer
 Dim str1 as String
 Dim value1 as Integer
 x=InputBox("Enter a value for x: ")
 y=InputBox("Enter a value for y: ")
 MsgBox "The sum of these numbers is: " & x+y
 str1=Str(x) & Str(y)
 MsgBox "The concatenated string for these numbers is: " & str1
End Sub
```

**See Also** `Format`  
`Val`

## StrComp

### Function

---

**Description** Compares two strings and returns an integer specifying the result of the comparison.

## StrComp

**Syntax** `StrComp(string1$, string2$[, compare%])`

| Syntax Element   | Description                                                                     |
|------------------|---------------------------------------------------------------------------------|
| <i>string1\$</i> | Any expression containing the first string to compare.                          |
| <i>string2\$</i> | The second string to compare.                                                   |
| <i>compare%</i>  | An integer for the method of comparison (0=case-sensitive, 1=case-insensitive). |

**Comments** StrComp returns one of the following values:

| Value | Meaning                                            |
|-------|----------------------------------------------------|
| -1    | <i>string1\$</i> < <i>string2\$</i>                |
| 0     | <i>string1\$</i> = <i>string2\$</i>                |
| >1    | <i>string1\$</i> > <i>string2\$</i>                |
| Null  | <i>string1\$</i> = Null or <i>string2\$</i> = Null |

If *compare%* is 0, a case sensitive comparison based on the ANSI character set sequence is performed. If *compare%* is 1, a case insensitive comparison is done based upon the relative order of characters as determined by the country code setting for your system. If omitted, the module level default, as specified with Option Compare is used.

The *string1* and *string2* arguments are both passed as variants. Therefore, any type of expression is supported. Numbers will be automatically converted to strings.

**Example** This example compares a user-entered string to the string Smith.

```
Option Compare Text
Sub main
 Dim lastname as String
 Dim smith as String
 Dim x as Integer
 smith="Smith"
 lastname=InputBox("Type your last name")
 x=StrComp(lastname,smith,1)
 If x=0 then
 MsgBox "You typed 'Smith' or 'smith'."
 Else
 MsgBox "You typed: " & lastname & " not 'Smith'."
 End If
End Sub
```

**See Also** Instr  
Option Compare

## String Function

---

**Description** Returns a string consisting of a repeated character.

**Syntax** Syntax A **String**[\$](*number*, *Character%*)

Syntax B **String**[\$](*number*, *string-expression*\$)

| Syntax Element              | Description                                                                                                         |
|-----------------------------|---------------------------------------------------------------------------------------------------------------------|
| \$                          | Optional. If specified the return type is String. If omitted, the function returns a Variant of VarType 8 (String). |
| <i>number</i>               | The length of the string to be returned.                                                                            |
| <i>Character%</i>           | A numeric expression that contains an integer for the decimal ANSI code of the character to use.                    |
| <i>string-expression</i> \$ | A string argument, the first character of which becomes the repeated character.                                     |

**Comments** *number* must be between 0 and 32,767.

*Character%* must evaluate to an integer between 0 and 255.

**Example** This example places asterisks (\*) in front of a string that is printed as a payment amount.

```
Sub main
 Dim str1 as String
 Dim size as Integer
 i: str1=InputBox("Enter an amount up to 999,999.99: ")
 If Instr(str1,".")=0 then str1 = str1 + ".00"
 If Len(str1)>10 then
 MsgBox "Amount too large. Try again."
 Goto i
 End If
 size=10-Len(str1)
 ' Print amount in a space on a check allotted for 10 characters
 str1=String(size,Asc("*")) & str1
 MsgBox "The amount is: $" & str1
End Sub
```

**See Also** Space  
Str

Sub...End Sub

## Sub...End Sub

Statement

---

Description Defines a sub procedure.

Syntax [Static] [Private] **Sub** *name* [(Optional) *arg* [As *type*],...]

### End Sub

| Syntax Element | Description                                                                                             |
|----------------|---------------------------------------------------------------------------------------------------------|
| <i>name</i>    | The name of the sub procedure.                                                                          |
| <i>arg</i>     | An argument to pass to the sub procedure when it is called. Multiple arguments are separated by commas. |
| <i>type</i>    | The data type of an argument in <i>arg</i> .                                                            |

Comments A call to a sub procedure stands alone as a separate statement. (See the `Call` statement). Recursion is supported.

*arg* contains an argument being passed to the sub procedure. An argument is represented by a variable name. Multiple arguments are separated by commas. Note the following information about the arguments being passed:

- ▶ The data type of an argument can be specified through a type declaration character or through the `As` clause.
- ▶ Arguments of a `User-Defined` data type are declared through an `As` clause and a *type* that has previously been defined through the `Type` statement.
- ▶ If an argument is an array, use empty parentheses after the argument name. The array dimensions are not specified within the `Sub` statement. All references to the array within the body of the sub procedure must have a consistent number of dimensions.
- ▶ If you declare an argument as `Optional`, its value can be omitted when the sub procedure is called. Only arguments with `Variant` data types can be declared as optional, and all optional arguments must appear after any required arguments in the `Sub` statement. Use the function `IsMissing` to check whether an optional argument was actually sent to the sub procedure or was omitted.
- ▶ Arguments can be listed in a particular order, or they can be identified by name. See the `Call` statement for information on named arguments.

The sub procedure returns to the caller when the `End Sub` statement is reached or when an `Exit Sub` statement is executed.

The `Static` keyword specifies that all the variables declared within the sub procedure will retain their values as long as the program is running, regardless of the way the variables are declared.

The `Private` keyword specifies that the procedures will not be accessible to functions and sub procedures from other modules. Only procedures defined in the same module will have access to a `Private` sub procedure.

SQABasic procedures use the call-by-reference convention by default. This means that if the called procedure changes the value of an argument passed in `arg`, the new value will apply in the calling procedure as well. This feature should be used with great care.

The `MAIN` sub procedure has a special meaning. In many implementations of Basic, `MAIN` will be called when the module is run. The `MAIN` sub procedure is not allowed to take arguments.

Use `Function` to define a procedure that has a return value.

**Example** This example is a sub procedure that uses the `Sub . . . End Sub` statement.

```
Sub main
 MsgBox "Hello, World."
End Sub
```

**See Also**

|              |                 |
|--------------|-----------------|
| Call         | Global          |
| Dim          | Option Explicit |
| Function     | Static          |
| End Function |                 |

## SysMenuIDSelect

User Action Command



**Description** Performs a system menu selection based on the internal ID of the menu item. A system menu is the menu that appears when you click on the control box in the upper-left corner of a window.

**Syntax** `SysMenuIDSelect MenuID&`

| Syntax Element           | Description                       |
|--------------------------|-----------------------------------|
| <code>MenuID&amp;</code> | The internal ID of the menu item. |

SysMenuSelect

**Comments** This command is necessary for making selections from System menu items that do not contain text, such as owner drawn or bitmap menus.

**Example** This example selects the menu item identified by the internal ID 2034 from the System menu of the current context window.

```
SysMenuIDSelect 2034
```

**See Also** MenuIDSelect                      PopupMenuSelect  
MenuSelect                            SysMenuIDSelect  
PopupMenuIDSelect

## SysMenuSelect

User Action Command



**Description** Perform a system menu selection based on the text of the menu item. A system menu is the menu that appears when you click on the control box in the upper-left corner of a window.

**Syntax** **SysMenuSelect** *menuPath\$*

| Syntax Element    | Description                |
|-------------------|----------------------------|
| <i>menuPath\$</i> | The name of the menu item. |

**Comments** The sub-menus are delimited by a pointer (->). Robot can recognize menus and sub-menus up to 5 levels deep.

**Example** This example selects the menu item Switch To... from the System menu of the current context window.

```
SysMenuSelect "Switch To..."
```

**See Also** MenuIDSelect                      PopupMenuSelect  
MenuSelect                            SysMenuSelect  
PopupMenuIDSelect



## Tab Function

---

**Description** Moves the current print position to the column specified.

**Syntax** **Tab** (*n*)

| Syntax Element | Description                    |
|----------------|--------------------------------|
| <i>n</i>       | The new print position to use. |

**Comments** The `Tab` function can be used only inside `Print` statement. The leftmost print position is position number 1.

When the `Print` statement is used, the `Tab` function uses the following rules for determining the next print position:

1. If *n* is less than the total line width, the new print position is *n*.
2. If *n* is greater than the total line width, the new print position is *n* Mod *width*.
3. If the current print position is greater than *n* or *n* Mod *width*, `Tab` skips to the next line and sets the print position to *n* or *n* Mod *width*.

To set the width of a print line, use the `Width` statement.

**Example** This example prints the octal values for the numbers from 1 to 25. It uses `Tab` to put five character spaces between the values.

```
Sub main
 Dim x as Integer
 Dim y
 For x=1 to 25
 y=Oct$(x)
 Print x Tab(10) y
 Next x
End Sub
```

**See Also** `Print`      `Spc`  
`Space`      `Width`

# TabControl

User Action Command

**Description**    Performs an action on a tab control.

**Syntax**            **TabControl** *action%*, *recMethod\$*, *parameters\$*

| Syntax Element     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i>     | <p>One of these actions:</p> <ul style="list-style-type: none"> <li>▶ <i>MouseClicked</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> must contain <i>Coords=x,y</i>.</li> <li>▶ <i>MouseDown</i>. The dragging of the mouse while mouse buttons and/or shifting keys (Ctrl, Alt, Shift) are pressed. When <i>action%</i> contains a mouse-drag value, <i>parameters\$</i> must contain <i>Coords=x1,y1,x2,y2</i>. See Appendix E for a list of mouse click and drag values.</li> <li>▶ <i>ScrollAction</i>. One of these scroll actions: <ul style="list-style-type: none"> <li>ScrollPageRight    ScrollPageDown</li> <li>ScrollRight        ScrollLineDown</li> <li>ScrollPageLeft    ScrollPageUp</li> <li>ScrollLeft         ScrollLineUp</li> <li>HScrollTo         VScrollTo</li> </ul> <p>HScrollTo and VScrollTo take the required parameter <i>Position=%</i>.</p> <p>If Robot cannot interpret the action being applied to a scroll bar, which happens with certain custom standalone scroll bars, it records the action as a click or drag.</p> </li> </ul> |
| <i>recMethod\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <i>ID=%</i>. The object's internal Windows ID.</li> <li>▶ <i>Index=%</i>. The number of the object among all objects identified with the same base recognition method. Typically, <i>Index</i> is used after another recognition method qualifier.</li> <li>▶ <i>ItemIndex=%</i>. The index of the tab item acted upon. Used only after one of these parent values: <i>ID=%</i>, <i>ObjectIndex=%</i>, <i>Name=\$</i>, <i>Text=\$</i>. Parent/child values are separated by a backslash and semicolons (<i>i\i</i>).</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |

» » »

▶ ▶ ▶

| Syntax Element      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                     | <ul style="list-style-type: none"> <li>▶ <code>ItemText=\$</code>. The text of the tab item acted upon. Used only after one of these parent values: <code>ID=%</code>, <code>ObjectIndex=%</code>, <code>Name=\$</code>, <code>Text=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>;\</code>).</li> <li>▶ <code>JavaText=\$</code>. A label that identifies the object in the user interface.</li> <li>▶ <code>Name=\$</code>. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a <code>command</code> button might be <code>Command1</code>.</li> <li>▶ <code>ObjectIndex=%</code>. The number of the object among all objects of the same type in the same window.</li> <li>▶ <code>State=\$</code>. An optional qualifier for any other recognition method. There are two possible values for this setting: <code>Enabled</code> and <code>Disabled</code>. The default state is the state of the current context window (as set in the most recent <code>Window SetContext</code> command), or <code>Enabled</code> if the state has not been otherwise declared.</li> <li>▶ <code>Text=\$</code>. The text displayed on the object.</li> <li>▶ <code>Type=\$</code>. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The <code>Type</code> qualifier uses the following form: <code>Type=\$;recMethod=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>;\</code>).</li> <li>▶ <code>VisualText=\$</code>. An optional setting used to identify an object by its prior label. It is for user clarification only and does not affect object recognition.</li> </ul> |
| <i>parameters\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>Coords=x,y</code>. If <code>action%</code> is a mouse click, specifies the <code>x,y</code> coordinates of the click, relative to the top left of the object or the item.</li> <li>▶ <code>Coords=x1,y1,x2,y2</code>. If <code>action%</code> is a mouse drag, specifies the coordinates, where <code>x1,y1</code> are the starting coordinates of the drag, and <code>x2,y2</code> are the ending coordinates. The coordinates are relative to the top left of the object or the item.</li> <li>▶ <code>Index=%</code>. In Java environments, specifies the index of the tab being acted upon.</li> <li>▶ <code>Position=%</code>. If <code>action%</code> is <code>VScrollTo</code> or <code>HScrollTo</code>, specifies the scroll bar value of the new scrolled-to position in the scroll box. Every scroll bar has an internal range, and this value is specific to that range.</li> <li>▶ <code>Text=\$</code>. In Java environments, specifies the label of the tab being acted upon.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Comments            | None.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

TabControlVP

**Example** This example clicks the item identified by the text `System` at `x,y` coordinates of `50,25` in the first tab control in the window (`ObjectIndex=1`). The clicked tab is labeled `System`.

```
TabControl Click, "ObjectIndex=1;\;ItemText=System", "Coords=50,25"
```

**See Also** TabControlVP

## TabControlVP

Verification Point Command

»»SQA»

**Description** Establishes a verification point for a tab control.

**Syntax** `Result = TabControlVP (action%, recMethod$, parameters$)`

| Syntax Element       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>action%</code> | <p>The type of verification to perform. Valid values:</p> <ul style="list-style-type: none"><li>▶ <code>CompareData</code>. Captures the data of the object and compares it to a recorded baseline. <code>parameters\$ VP</code> and <code>Type</code> are required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li><li>▶ <code>CompareNumeric</code>. Captures the numeric value of the text of the object and compares it to the value of <code>parameters\$ Value</code> or <code>Range</code>. <code>parameters\$ VP</code> and either <code>Value</code> or <code>Range</code> are required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li><li>▶ <code>CompareProperties</code>. Captures object properties information for the object and compares it to a recorded baseline. <code>parameters\$ VP</code> is required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li><li>▶ <code>CompareText</code>. Captures the text of the object and compares it to a recorded baseline. <code>parameters\$ VP</code> and <code>Type</code> are required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li></ul> |

▶ ▶ ▶

▶ ▶ ▶

| Syntax Element       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>recMethod</i> \$  | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>ID=%</code>. The object's internal Windows ID.</li> <li>▶ <code>Index=%</code>. The number of the object among all objects identified with the same base recognition method. Typically, <code>Index</code> is used after another recognition method qualifier.</li> <li>▶ <code>ItemIndex=%</code>. The index of the tab item acted upon. Used only after one of these parent values: <code>ID=%</code>, <code>ObjectIndex=%</code>, <code>Name=\$</code>, <code>Text=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>i \ i</code>).</li> <li>▶ <code>ItemText=\$</code>. The text of the tab item acted upon. Used only after one of these parent values: <code>ID=%</code>, <code>ObjectIndex=%</code>, <code>Name=\$</code>, <code>Text=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>i \ i</code>).</li> <li>▶ <code>JavaText=\$</code>. A label that identifies the object in the user interface.</li> <li>▶ <code>Name=\$</code>. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be <code>Command1</code>.</li> <li>▶ <code>ObjectIndex=%</code>. The number of the object among all objects of the same type in the same window.</li> <li>▶ <code>Text=\$</code>. The text displayed on the object.</li> <li>▶ <code>Type=\$</code>. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The <code>TYPE</code> qualifier uses the following form: <code>Type=\$ ; recMethod=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>i \ i</code>).</li> </ul> |
| <i>parameters</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>ExpectedResult=%</code>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> <li>– <code>PASS</code>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail.</li> <li>– <code>FAIL</code>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail.</li> </ul> </li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

▶ ▶ ▶

## TabControlVP

▶ ▶ ▶

| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                | <ul style="list-style-type: none"> <li>▶ <code>Range=&amp; , &amp;</code>. Used with the action <code>CompareNumeric</code> when a numeric range comparison is being performed, as in <code>Range=2 , 12</code> (test for numbers in this range). The values are inclusive.</li> <li>▶ <code>Type=\$</code>. Specifies the verification method to use for <code>CompareText</code> actions. The possible values are: <code>CaseSensitive</code>, <code>CaseInsensitive</code>, <code>FindSubStr</code>, <code>FindSubStrI</code> (case insensitive), and <code>UserDefined</code>. See <i>Comments</i> for more information. If <code>UserDefined</code> is specified, two additional parameters are required: <ul style="list-style-type: none"> <li>- <code>DLL=\$</code>. The full path and file name of the library that contains the function</li> <li>- <code>Function=\$</code>. The name of the custom function to use in comparing the text</li> </ul> </li> <li>▶ <code>Value=&amp;</code>. Used with the action <code>CompareNumeric</code> when a numeric equivalence comparison is being performed, as in <code>Value=25</code> (test against the value 25).</li> <li>▶ <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points.</li> <li>▶ <code>Wait=% , %</code>. A <code>Wait State</code> that specifies the verification point's <code>Retry</code> value and a <code>Timeout</code> value, as in <code>Wait=10 , 40</code> (retry the test every 10 seconds, but time out the test after 40 seconds).</li> </ul> |

**Comments** This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

With the `Type=$` parameter, `CaseSensitive` and `CaseInsensitive` require a full match between the current baseline text and the text captured during playback. With `FindSubStr` and `FindSubStrI`, the current baseline can be a substring of the text captured during playback. The substring can appear anywhere in the playback text. To modify the current baseline text, double-click the verification point name in the Robot Asset pane (to the left of the script).

**Example** This example captures the properties of the first tab control in the window (`ObjectIndex=1`) and compares them to the recorded baseline in verification point TEST1A.

```
Result=TabControlVP(CompareProperties,"ObjectIndex=1","VP=TEST1A")
```

**See Also** `TabControl`

## Tan

### Function

---

**Description** Returns the tangent of an angle in radians.

**Syntax** **Tan**(*number*)

| Syntax Element | Description                                    |
|----------------|------------------------------------------------|
| <i>number</i>  | An expression containing the angle in radians. |

**Comments** *number* is specified in radians, and can be either positive or negative.

The return value is single-precision if the angle is an integer, currency or single-precision value, double precision for a long, Variant or double-precision value.

To convert degrees to radians, multiply by PI/180. The value of PI is 3.14159.

**Example** This example finds the height of the exterior wall of a building, given its roof pitch and the length of the building.

```
Sub main
 Dim bldglen, wallht
 Dim pitch
 Dim msgtext
 Const PI=3.14159
 Const conversion= PI/180
 On Error Resume Next
 pitch=InputBox("Enter the roof pitch in degrees:")
 pitch=pitch*conversion
 bldglen=InputBox("Enter the length of the building in feet:")
 wallht=Tan(pitch)*(bldglen/2)
 msgtext="The building height is: " & Format(wallht,"##.00")
 MsgBox msgtext
End Sub
```

**See Also** Atn  
Cos  
Sin  
Derived Trigonometric functions (Appendix D)

Text

## Text Statement

---

Description Places line(s) of text in a dialog box.

Syntax **Text** *x, y, dx, dy, text\$[, .id]*

| Syntax Element | Description                                                                                              |
|----------------|----------------------------------------------------------------------------------------------------------|
| <i>x, y</i>    | The upper left corner coordinates of the text area, relative to the upper left corner of the dialog box. |
| <i>dx, dy</i>  | The width and height of the text area.                                                                   |
| <i>text\$</i>  | A string containing the text to appear in the text area defined by <i>x, y</i> .                         |
| <i>.id</i>     | An optional identifier used by the dialog statements that act on this control.                           |

Comments If the width of *text\$* is greater than *dx*, the spillover characters wrap to the next line. This will continue as long as the height of the text area established by *dy* is not exceeded. Excess characters are truncated.

By preceding an underlined character in *text\$* with an ampersand (&), you enable a user to press the underlined character on the keyboard and position the cursor in the combo or text box defined in the statement immediately following the `Text` statement.

Use the `Text` statement only between a `Begin Dialog` and an `End Dialog` statement.

Example This example defines a dialog box with a combination list and text box and three buttons.

```
Sub main
 Dim ComboBox1() as String
 ReDim ComboBox1(0)
 ComboBox1(0)=Dir("C:*. *")
 Begin Dialog UserDialog 166, 142, "SQABasic Dialog Box"
 Text 9, 3, 69, 13, "Filename:", .Text1
 DropComboBox 9, 14, 81, 119, ComboBox1(), .ComboBox1
 OKButton 101, 6, 54, 14
 CancelButton 101, 26, 54, 14
 PushButton 101, 52, 54, 14, "Help", .Push1
 End Dialog
 Dim mydialog as UserDialog
 On Error Resume Next
 Dialog mydialog
 If Err=102 then MsgBox "Dialog box canceled."
End Sub
```



|          |              |              |                |
|----------|--------------|--------------|----------------|
| See Also | Begin Dialog | CheckBox     | OKButton       |
|          | End Dialog   | ComboBox     | OptionButton   |
|          | Button       | Dialog       | OptionGroup    |
|          | ButtonGroup  | DropComboBox | Picture        |
|          | CancelButton | GroupBox     | StaticComboBox |
|          | Caption      | ListBox      | TextBox        |

## TextBox

### Statement

---

**Description** Creates a text box in a dialog box.

**Syntax** **TextBox** [NoEcho] *x, y, dx, dy, .field*

| Syntax Element | Description                                                                                             |
|----------------|---------------------------------------------------------------------------------------------------------|
| <i>x, y</i>    | The upper left corner coordinates of the text box, relative to the upper left corner of the dialog box. |
| <i>dx, dy</i>  | The width and height of the text box area.                                                              |
| <i>.field</i>  | The name of the dialog-record field to hold the text string.                                            |

**Comments** A *dy* value of 12 will usually accommodate text in the system font.

When the user selects the OK button, or any PushButton other than cancel, the text string entered in the text box will be recorded in *.field*.

The `NoEcho` keyword is often used for passwords; it displays all characters entered as asterisks (\*).

Use the `TextBox` statement only between a `Begin Dialog` and an `End Dialog` statement.

**Example** This example creates a dialog box with a group box, and two buttons.

```

Sub main
 Begin Dialog UserDialog 194, 76, "SQABasic Dialog Box"
 GroupBox 9, 8, 97, 57, "File Range"
 OptionGroup .OptionGroup2
 OptionButton 19, 16, 46, 12, "All pages", .OptionButton3
 OptionButton 19, 32, 67, 8, "Range of pages", .OptionButton4
 Text 25, 43, 20, 10, "From:", .Text6
 Text 63, 43, 14, 9, "To:", .Text7
 TextBox 79, 43, 13, 12, .TextBox4
 TextBox 47, 43, 12, 11, .TextBox5
 OKButton 135, 6, 54, 14
 CancelButton 135, 26, 54, 14
 End Dialog

```

## Time (Function)

```
Dim mydialog as UserDialog
On Error Resume Next
Dialog mydialog
If Err=102 then
 MsgBox "Dialog box canceled."
End If
End Sub
```

|          |              |              |                |
|----------|--------------|--------------|----------------|
| See Also | Begin Dialog | CheckBox     | OKButton       |
|          | End Dialog   | ComboBox     | OptionButton   |
|          | Button       | Dialog       | OptionGroup    |
|          | ButtonGroup  | DropComboBox | Picture        |
|          | CancelButton | GroupBox     | StaticComboBox |
|          | Caption      | ListBox      | Text           |
|          |              |              |                |

## Time Function

---

**Description** Returns a string representing the current time.

**Syntax** **Time**[ \$ ]

| Syntax Element | Description                                                                                                          |
|----------------|----------------------------------------------------------------------------------------------------------------------|
| \$             | Optional. If specified, the return type is String. If omitted, the function returns a Variant of VarType 8 (String). |

**Comments** The **Time** function returns an eight character string. The format of the string is *hh:mm:ss* where *hh* is the hour, *mm* is the minutes and *ss* is the seconds. The hour is specified in military style, and ranges from 0 to 23.

**Example** This example writes data to a file if it hasn't been saved within the last 2 minutes.

```
Sub main
Dim tempfile
Dim filetime, curtime
Dim msgtext
Dim acctno(100) as Single
Dim x, I
tempfile="C:\TEMP001"
Open tempfile For Output As #1
filetime=FileDateTime(tempfile)
x=1
I=1
acctno(x)=0
Do
 curtime=Time
 acctno(x)=InputBox("Enter an account number (99 to end):")
 If acctno(x)=99 then
 For I=1 to x-1
 Write #1, acctno(I)
 Next I
 End If
 Exit Do

```

```

ElseIf (Minute(filetime)+2)<=Minute(curtime) then
 For I=I to x
 Write #1, acctno(I)
 Next I
End If
x=x+1
Loop
Close #1
x=1
msgtext="Contents of C:\TEMP001 is:" & Chr(10)
Open tempfile for Input as #1
Do While Eof(1)<>-1
 Input #1, acctno(x)
 msgtext=msgtext & Chr(10) & acctno(x)
 x=x+1
Loop
MsgBox msgtext
Close #1
Kill "C:\TEMP001"
End Sub

```

See Also

|                |            |
|----------------|------------|
| Date function  | Timer      |
| Date statement | TimeSerial |
| Time statement | TimeValue  |

## Time Statement

---

**Description** Sets the system time.

**Syntax** **Time** = *expression*

| Syntax Element    | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>expression</i> | <p>An expression that evaluates to a valid time. When <b>Time</b> (with the dollar sign \$) is used, the <i>expression</i> must evaluate to a string of one of the following forms:</p> <ul style="list-style-type: none"> <li>▶ <i>hh</i>. Set the time to <i>hh</i> hours 0 minutes and 0 seconds.</li> <li>▶ <i>hh:mm</i>. Set the time to <i>hh</i> hours <i>mm</i> minutes and 0 seconds.</li> <li>▶ <i>hh:mm:ss</i>. Set the time to <i>hh</i> hours <i>mm</i> minutes and <i>ss</i> seconds.</li> </ul> |

**Comments** **Time** uses a 24-hour clock. Thus, 6:00 P.M. must be entered as 18:00:00.

If *expression* is not already a Variant of VarType 7 (date), **Time** attempts to convert it to a valid time. It recognizes time separator characters defined in the International section of the Windows Control Panel. **Time** (without the \$) accepts both 12 and 24 hour clocks.

Timer

**Example** This example changes the time on the system clock.

```
Sub main
 Dim newtime, answer as String
 On Error Resume Next
i: newtime=InputBox("What time is it?")
 answer=InputBox("Is this AM or PM?")
 If answer="PM" or answer="pm" then newtime=newtime & "PM"
 Time=newtime
 If Err<>0 then
 MsgBox "Invalid time. Try again."
 Err=0
 Goto i
 End If
End Sub
```

**See Also** Date function      TimeSerial  
Date statement      TimeValue  
Time function

## Timer Function

---

**Description** Returns the number of seconds that have elapsed since midnight.

**Syntax**      **Timer**

**Comments** The **Timer** function can be used in conjunction with the **Randomize** statement to seed the random number generator.

**Example** This example uses the **Timer** function to find a Megabucks number.

```
Sub main
 Dim msgtext, nextvalue, x, y
 Dim value(9)
 msgtext="Your Megabucks numbers are: "
 For x = 1 to 8
 Do
 value(x)=Timer
 value(x)=value(x)*100
 value(x)=Str(value(x))
 value(x)=Val(Right(value(x),2))
 Loop Until value(x)>1 and value(x)<36
 For y=1 to 1500
 Next y
 Next x
 For y = 1 to 8
 For x = 1 to 8
 If y <> x then
 If value(y)=value(x) then value(x)=value(x)+1
 End If
 Next x
 Next y
 For x = 1 to 8
 msgtext=msgtext & value(x) & " "
 Next x
 MsgBox msgtext
End Sub
```

**See Also** Randomize

# TimeSerial

## Function

---

**Description** Returns a time as a Variant of type 7 (date/time) for a specific hour, minute, and second.

**Syntax** **TimeSerial**(*hour%*, *minute%*, *second%* )

| Syntax Element | Description                               |
|----------------|-------------------------------------------|
| <i>hour%</i>   | A numeric expression for an hour (0-23).  |
| <i>minute%</i> | A numeric expression for a minute (0-59). |
| <i>second%</i> | A numeric expression for a second (0-59). |

**Comments** You also can specify relative times for each argument by using a numeric expression representing the number of hours, minutes, or seconds before or after a certain time.

**Example** This example displays the current time using TimeSerial.

```
Sub main
 Dim y
 Dim msgtext
 Dim nowhr
 Dim nowmin
 Dim nowsec
 nowhr=Hour(Now)
 nowmin=Minute(Now)
 nowsec=Second(Now)
 y=TimeSerial(nowhr,nowmin,nowsec)
 msgtext="The time is: " & y
 MsgBox msgtext
End Sub
```

**See Also**

|            |           |
|------------|-----------|
| DateSerial | Now       |
| Date Value | Second    |
| Hour       | TimeValue |
| Minute     |           |

## TimeValue

### Function

---

**Description** Returns a time value for a specified string.

**Syntax** **TimeValue**(*time\$*)

| Syntax Element | Description              |
|----------------|--------------------------|
| <i>time\$</i>  | A valid date time value. |

**Comments** The **TimeValue** function returns a Variant of `VarType 7 (date/time)` that represents a time between 0:00:00 and 23:59:59, or 12:00:00 A.M. and 11:59:59 P.M., inclusive.

**Example** This example writes a variable to a disk file based on a comparison of its last saved time and the current time. Note that all the variables used for the **TimeValue** function are dimensioned as `Double`, so that calculations based on their values will work properly.

```
Sub main
 Dim tempfile
 Dim ftime
 Dim filetime as Double
 Dim curtime as Double
 Dim minutes as Double
 Dim acctno(100) as Integer
 Dim x, I
 Dim msgtext as String
 tempfile="C:\TEMP001"
 Open tempfile For Output As 1
 ftime=FileDateTime(tempfile)
 filetime=TimeValue(ftime)
 minutes= TimeValue("00:02:00")
 x=1
 I=1
 acctno(x)=0
 Do
 curtime= TimeValue(Time)
 acctno(x)=InputBox("Enter an account number (99 to end):")
 If acctno(x)=99 then
 For I=I to x-1
 Write #1, acctno(I)
 Next I
 Exit Do
 ElseIf filetime+minutes<=curtime then
 For I=I to x
 Write #1, acctno(I)
 Next I
 End If
 x=x+1
 Loop
 Close #1
```

```
x=1
msgtext="You entered:" & Chr(10)
Open tempfile for Input as #1
Do While Eof(1)<>-1
 Input #1, acctno(x)
 msgtext=msgtext & Chr(10) & acctno(x)
 x=x+1
Loop
MsgBox msgtext
Close #1
Kill "C:\TEMP001"
End Sub
```

See Also      DateSerial          Now  
                  Date Value        Second  
                  Hour                TimeSerial  
                  Minute

## Toolbar

### User Action Command

**Description**      Performs an action on a toolbar control.

**Syntax**            **Toolbar** *action%*, *recMethod\$*, *parameters\$*

| Syntax Element     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i>     | <p>One of these mouse actions:</p> <ul style="list-style-type: none"> <li>▶ <i>MouseClicked</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> must contain <i>Coords=x,y</i>.</li> <li>▶ <i>MouseDown</i>. The dragging of the mouse while mouse buttons and/or shifting keys (Ctrl, Alt, Shift) are pressed. When <i>action%</i> contains a mouse-drag value, <i>parameters\$</i> must contain <i>Coords=x1,y1,x2,y2</i>.</li> </ul> <p>See Appendix E for a list of mouse click and drag values.</p> |
| <i>recMethod\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <i>ID=%</i>. The object's internal Windows ID.</li> <li>▶ <i>Name=\$</i>. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be <i>Command1</i>.</li> <li>▶ <i>ObjectIndex=%</i>. The number of the object among all objects of the same type in the same window.</li> </ul>                                                                                                                                                                                                               |

▶ ▶ ▶

## ToolbarVP

▶ ▶ ▶

| Syntax Element            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|---------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                           | <ul style="list-style-type: none"><li>▶ <code>State=\$</code>. An optional qualifier for any other recognition method. There are two possible values for this setting: <code>Enabled</code> and <code>Disabled</code>. The default state is the state of the current context window (as set in the most recent <code>Window SetContext</code> command), or <code>Enabled</code> if the state has not been otherwise declared.</li><li>▶ <code>Text=\$</code>. The text displayed on the object.</li><li>▶ <code>VisualText=\$</code>. An optional setting used to identify an object by its visible text. It is for user clarification only and does not affect object recognition.</li></ul> |
| <code>parameters\$</code> | <p>Valid values:</p> <ul style="list-style-type: none"><li>▶ <code>Coords=x,y</code>. If <code>action%</code> is a mouse click, specifies the coordinates of the click, relative to the top left of the object.</li><li>▶ <code>Coords=x1,y1,x2,y2</code>. If <code>action%</code> is a mouse drag, specifies the coordinates, where <code>x1,y1</code> are the starting coordinates of the drag, and <code>x2,y2</code> are the ending coordinates. The coordinates are relative to the top left of the object.</li></ul>                                                                                                                                                                    |

Comments None.

Example This example clicks the item identified by the text `System` at `x,y` coordinates of `50,25` in the first toolbar control in the window (`ObjectIndex=1`).

```
Toolbar Click, "ObjectIndex=1;ItemText=System", "Coords=50,25"
```

See Also `ToolbarVP`

## ToolbarVP

Verification Point Command

▶▶SQA▶

Description Establishes a verification point for a toolbar control.

Syntax `Result = ToolbarVP (action%, recMethod$, parameters$)`



| Syntax Element      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i>      | <p>The type of verification to perform. Valid values:</p> <ul style="list-style-type: none"> <li>▶ <i>CompareNumeric</i>. Captures the numeric value of the text of the object and compares it to the value of <i>parameters\$ Value</i> or <i>Range</i>. <i>parameters\$ VP</i> and either <i>Value</i> or <i>Range</i> are required; <i>ExpectedResult</i> and <i>Wait</i> are optional.</li> <li>▶ <i>CompareProperties</i>. Captures object properties information for the object and compares it to a recorded baseline. <i>parameters\$ VP</i> is required; <i>ExpectedResult</i> and <i>Wait</i> are optional.</li> <li>▶ <i>CompareText</i>. Captures the text of the object and compares it to a recorded baseline. <i>parameters\$ VP</i> and <i>Type</i> are required; <i>ExpectedResult</i> and <i>Wait</i> are optional.</li> </ul>                                                                                                                                                                                                                                                                                                                                                          |
| <i>recMethod\$</i>  | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <i>ID=%</i>. The object's internal Windows ID.</li> <li>▶ <i>Name=\$</i>. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be <i>Command1</i>.</li> <li>▶ <i>ObjectIndex=%</i>. The number of the object among all objects of the same type in the same window.</li> <li>▶ <i>Text=\$</i>. The text displayed on the object.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <i>parameters\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <i>ExpectedResult=%</i>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> <li>– <i>PASS</i>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail.</li> <li>– <i>FAIL</i>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail.</li> </ul> </li> <li>▶ <i>Range=&amp; , &amp;</i>. Used with the action <i>CompareNumeric</i> when a numeric range comparison is being performed, as in <i>Range=2 , 12</i> (test for numbers in this range). The values are inclusive.</li> <li>▶ <i>Type=\$</i>. Specifies the verification method to use for <i>CompareText</i> actions. The possible values are: <i>CaseSensitive</i>, <i>CaseInsensitive</i>, <i>FindSubStr</i>, <i>FindSubStrI</i> (case insensitive), and <i>UserDefined</i>. See <i>Comments</i> for more information.</li> </ul> |

▶ ▶ ▶

## ToolbarVP

▶ ▶ ▶

| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                | <p>If <code>UserDefined</code> is specified, two additional parameters are required:</p> <ul style="list-style-type: none"><li>- <code>DLL=\$</code>. The full path and file name of the library that contains the function</li><li>- <code>Function=\$</code>. The name of the custom function to use in comparing the text</li></ul> <p>▶ <code>Value=&amp;</code>. Used with the action <code>CompareNumeric</code> when a numeric equivalence comparison is being performed, as in <code>Value=25</code> (test against the value 25).</p> <p>▶ <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points.</p> <p>▶ <code>Wait=% , %</code>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10 , 40</code> (retry the test every 10 seconds, but time out the test after 40 seconds).</p> |

**Comments** This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

With the `Type=$` parameter, `CaseSensitive` and `CaseInsensitive` require a full match between the current baseline text and the text captured during playback. With `FindSubStr` and `FindSubStrI`, the current baseline can be a substring of the text captured during playback. The substring can appear anywhere in the playback text. To modify the current baseline text, double-click the verification point name in the Robot Asset pane (to the left of the script).

**Example** This example captures the properties of the first toolbar control in the window (`ObjectIndex=1`) and compares them to the recorded baseline in verification point `TEST1A`.

```
Result = ToolbarVP (CompareProperties, "ObjectIndex=1", "VP=TEST1A")
```

**See Also** [Toolbar](#)

## Trackbar

User Action Command

▶▶SQA▶

Description Performs an action on a trackbar control.

Syntax **Trackbar** *action%*, *recMethod\$*, *parameters\$*

| Syntax Element     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i>     | <p>One of these mouse actions:</p> <ul style="list-style-type: none"> <li>▶ <i>MouseClicked</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> must contain <i>Coords=x,y</i>.</li> <li>▶ <i>MouseDown</i>. The dragging of the mouse while mouse buttons and/or shifting keys (Ctrl, Alt, Shift) are pressed. When <i>action%</i> contains a mouse-drag value, <i>parameters\$</i> must contain <i>Coords=x1,y1,x2,y2</i>. See Appendix E for a list of mouse click and drag values.</li> <li>▶ <i>ScrollAction</i>. One of these scroll actions: <ul style="list-style-type: none"> <li>ScrollPageRight      ScrollPageDown</li> <li>ScrollRight          ScrollLineDown</li> <li>ScrollPageLeft      ScrollPageUp</li> <li>ScrollLeft            ScrollLineUp</li> <li>HScrollTo            VScrollTo</li> </ul> <p>HScrollTo and VScrollTo take the required parameter <i>Position=%</i>.</p> </li> </ul> <p>If Robot cannot interpret the action being applied to a trackbar, which happens with certain custom standalone trackbars, it records the action as a click or drag.</p> |
| <i>recMethod\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <i>ID=%</i>. The object's internal Windows ID.</li> <li>▶ <i>Index=%</i>. The number of the object among all objects identified with the same base recognition method. Typically, <i>Index</i> is used after another recognition method qualifier — for example, <i>Name=\$; Index=%</i>.</li> <li>▶ <i>JavaText=\$</i>. A label that identifies the object in the user interface.</li> <li>▶ <i>Name=\$</i>. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be <i>Command1</i>.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |

▶ ▶ ▶

## Trackbar

▶ ▶ ▶

| Syntax Element      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|---------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                     | <ul style="list-style-type: none"> <li>▶ <code>ObjectIndex=%</code>. The number of the object among all objects of the same type in the same window.</li> <li>▶ <code>State=\$</code>. An optional qualifier for any other recognition method. There are two possible values for this setting: <code>Enabled</code> and <code>Disabled</code>. The default state is the state of the current context window (as set in the most recent <code>Window SetContext</code> command), or <code>Enabled</code> if the state has not been otherwise declared.</li> <li>▶ <code>Text=\$</code>. The text displayed on the object.</li> <li>▶ <code>Type=\$</code>. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The <code>Type</code> qualifier uses the following form: <code>Type=\$;recMethod=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>;\;</code>).</li> <li>▶ <code>VisualText=\$</code>. An optional setting used to identify an object by its visible text. It is for user clarification only and does not affect object recognition.</li> </ul> |
| <i>parameters\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>Coords=x,y</code>. If <i>action%</i> is a mouse click, specifies the coordinates of the click, relative to the top left of the object.</li> <li>▶ <code>Coords=x1,y1,x2,y2</code>. If <i>action%</i> is a mouse drag, specifies the coordinates, where <i>x1,y1</i> are the starting coordinates of the drag, and <i>x2,y2</i> are the ending coordinates. The coordinates are relative to the top left of the object.</li> <li>▶ <code>Position=%</code>. If <i>action%</i> is <code>VScrollTo</code> or <code>HScrollTo</code>, specifies the scroll bar value of the new scrolled-to position. Every trackbar has an internal range and this parameter value is specific to that range.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                      |

Comments None.

Example This example clicks the item identified by the text `System` at *x,y* coordinates of 50,25 in the first trackbar control in the window (`ObjectIndex=1`).

```
Trackbar Click, "ObjectIndex=1;ItemText=System", "Coords=50,25"
```

See Also `TrackbarVP`

# TrackbarVP

Verification Point Command

»»SQA»

**Description** Establishes a verification point for a trackbar control.

**Syntax** `Result = TrackbarVP (action%, recMethod$, parameters$)`

| Syntax Element           | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>action%</code>     | <p>The type of verification to perform. Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>CompareNumeric</code>. Captures the numeric value of the text of the object and compares it to the value of <code>parameters\$ Value or Range</code>. <code>parameters\$ VP</code> and either <code>Value or Range</code> are required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li> <li>▶ <code>CompareProperties</code>. Captures object properties information for the object and compares it to a recorded baseline. <code>parameters\$ VP</code> is required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li> <li>▶ <code>CompareText</code>. Captures the text of the object and compares it to a recorded baseline. <code>parameters\$ VP</code> and <code>Type</code> are required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li> </ul> |
| <code>recMethod\$</code> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>ID=%</code>. The object's internal Windows ID.</li> <li>▶ <code>Index=%</code>. The number of the object among all objects identified with the same base recognition method. Typically, <code>Index</code> is used after another recognition method qualifier — for example, <code>Name=\$; Index=%</code>.</li> <li>▶ <code>JavaText=\$</code>. A label that identifies the object in the user interface.</li> <li>▶ <code>Name=\$</code>. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be <code>Command1</code>.</li> <li>▶ <code>ObjectIndex=%</code>. The number of the object among all objects of the same type in the same window.</li> <li>▶ <code>Text=\$</code>. The text displayed on the object.</li> </ul>          |

▶ ▶ ▶

▶ ▶ ▶

| Syntax Element            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|---------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                           | <ul style="list-style-type: none"> <li>▶ <code>Type=\$</code>. An optional qualifier for recognition methods. Used to identify the object within a specific context or environment. The <code>Type</code> qualifier uses the following form: <code>Type=\$;recMethod=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>;\;</code>).</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <code>parameters\$</code> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>ExpectedResult=%</code>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> <li>- <code>PASS</code>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail.</li> <li>- <code>FAIL</code>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail.</li> </ul> </li> <li>▶ <code>Range=&amp;, &amp;</code>. Used with the action <code>CompareNumeric</code> when a numeric range comparison is being performed, as in <code>Range=2, 12</code> (test for numbers in this range). The values are inclusive.</li> <li>▶ <code>Type=\$</code>. Specifies the verification method to use for <code>CompareText</code> actions. The possible values are: <code>CaseSensitive</code>, <code>CaseInsensitive</code>, <code>FindSubStr</code>, <code>FindSubStrI</code> (case insensitive), and <code>UserDefined</code>. See <i>Comments</i> for more information. If <code>UserDefined</code> is specified, two additional parameters are required: <ul style="list-style-type: none"> <li>- <code>DLL=\$</code>. The full path and file name of the library that contains the function</li> <li>- <code>Function=\$</code>. The name of the custom function to use in comparing the text</li> </ul> </li> <li>▶ <code>Value=&amp;</code>. Used with the action <code>CompareNumeric</code> when a numeric equivalence comparison is being performed, as in <code>Value=25</code> (test against the value 25).</li> <li>▶ <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points.</li> <li>▶ <code>Wait=%, %</code>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10, 40</code> (retry the test every 10 seconds, but time out the test after 40 seconds).</li> </ul> |

**Comments** This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

With the `Type=$` parameter, `CaseSensitive` and `CaseInsensitive` require a full match between the current baseline text and the text captured during playback. With `FindSubStr` and `FindSubStrI`, the current baseline can be a substring of the text captured during playback. The substring can appear anywhere in the playback text. To modify the current baseline text, double-click the verification point name in the Robot Asset pane (to the left of the script).

**Example** This example captures the properties of the first trackbar control in the window (`ObjectIndex=1`) and compares them to the recorded baseline in verification point `TEST1A`.

```
Result = TrackbarVP(CompareProperties,"ObjectIndex=1","VP=TEST1A")
```

**See Also** `Trackbar`

## TreeView

### User Action Command



**Description** Performs an action on a tree view control.

**Syntax** `TreeView action%, recMethod$, parameters$`

| Syntax Element               | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                              |                             |                          |                             |                             |                           |                         |                           |                        |                        |
|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------|-----------------------------|--------------------------|-----------------------------|-----------------------------|---------------------------|-------------------------|---------------------------|------------------------|------------------------|
| <code><i>action%</i></code>  | <p>One of these actions:</p> <ul style="list-style-type: none"> <li>▶ <i>MouseClicked</i>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> must contain <code>Coords=x,y</code>.</li> <li>▶ <i>MouseDown</i>. The dragging of the mouse while mouse buttons and/or shifting keys (Ctrl, Alt, Shift) are pressed. When <i>action%</i> contains a mouse-drag value, <i>parameters\$</i> must contain <code>Coords=x1,y1,x2,y2</code>. See Appendix E for a list of mouse click and drag values.</li> <li>▶ <i>ScrollAction</i>. One of these scroll actions: <table border="0" style="margin-left: 20px;"> <tr> <td><code>ScrollPageRight</code></td> <td><code>ScrollPageDown</code></td> </tr> <tr> <td><code>ScrollRight</code></td> <td><code>ScrollLineDown</code></td> </tr> <tr> <td><code>ScrollPageLeft</code></td> <td><code>ScrollPageUp</code></td> </tr> <tr> <td><code>ScrollLeft</code></td> <td><code>ScrollLineUp</code></td> </tr> <tr> <td><code>HScrollTo</code></td> <td><code>VScrollTo</code></td> </tr> </table> </li> </ul> | <code>ScrollPageRight</code> | <code>ScrollPageDown</code> | <code>ScrollRight</code> | <code>ScrollLineDown</code> | <code>ScrollPageLeft</code> | <code>ScrollPageUp</code> | <code>ScrollLeft</code> | <code>ScrollLineUp</code> | <code>HScrollTo</code> | <code>VScrollTo</code> |
| <code>ScrollPageRight</code> | <code>ScrollPageDown</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                              |                             |                          |                             |                             |                           |                         |                           |                        |                        |
| <code>ScrollRight</code>     | <code>ScrollLineDown</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                              |                             |                          |                             |                             |                           |                         |                           |                        |                        |
| <code>ScrollPageLeft</code>  | <code>ScrollPageUp</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                              |                             |                          |                             |                             |                           |                         |                           |                        |                        |
| <code>ScrollLeft</code>      | <code>ScrollLineUp</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                              |                             |                          |                             |                             |                           |                         |                           |                        |                        |
| <code>HScrollTo</code>       | <code>VScrollTo</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |                              |                             |                          |                             |                             |                           |                         |                           |                        |                        |



TreeView

▶ ▶ ▶

| Syntax Element      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                     | <p><code>HScrollTo</code> and <code>VScrollTo</code> take the required parameter <code>Position=%</code>.</p> <p>If Robot cannot interpret the action being applied to a scroll bar, which happens with certain custom standalone scroll bars, it records the action as a click or drag.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <i>recMethod\$</i>  | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>ID=%</code>. The object's internal Windows ID.</li> <li>▶ <code>ItemIndex=%</code>. The index of the tree view item acted upon. Used only after one of these parent values: <code>ID=%</code>, <code>ObjectIndex=%</code>, <code>Name=\$</code>, <code>Text=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>;\;</code>).</li> <li>▶ <code>ItemText=\$</code>. The text of the tree view item acted upon. Used only after one of these parent values: <code>ID=%</code>, <code>ObjectIndex=%</code>, <code>Name=\$</code>, <code>Text=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>;\;</code>).</li> <li>▶ <code>Name=\$</code>. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be <code>Command1</code>.</li> <li>▶ <code>ObjectIndex=%</code>. The number of the object among all objects of the same type in the same window.</li> <li>▶ <code>State=\$</code>. An optional qualifier for any other recognition method. There are two possible values for this setting: <code>Enabled</code> and <code>Disabled</code>. The default state is the state of the current context window (as set in the most recent <code>Window SetContext</code> command), or <code>Enabled</code> if the state has not been otherwise declared.</li> <li>▶ <code>Text=\$</code>. The text displayed on the object.</li> <li>▶ <code>VisualText=\$</code>. An optional setting used to identify an object by its prior label. It is for user clarification only and does not affect object recognition.</li> </ul> |
| <i>parameters\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>Coords=x,y</code>. If <i>action%</i> is a mouse click, specifies the <i>x,y</i> coordinates of the click, relative to the top left of the object or the item.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

▶ ▶ ▶





| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                | <ul style="list-style-type: none"> <li>▶ <code>Coords=x1,y1,x2,y2</code>. If <code>action%</code> is a mouse drag, specifies the coordinates, where <code>x1,y1</code> are the starting coordinates of the drag, and <code>x2,y2</code> are the ending coordinates. The coordinates are relative to the top left of the object or the item.</li> <li>▶ <code>Location=\$</code>. The part of the tree where the click occurred. Valid values:               <ul style="list-style-type: none"> <li>- <code>Text</code> or <code>Label</code> (the default). The text displayed on the clicked item.</li> <li>- <code>Button</code>. The plus or minus sign used to expand or collapse branches of the tree.</li> <li>- <code>Icon</code>. The icon displayed on the clicked item.</li> <li>- <code>StateIcon</code>. The icon that displays the state of the tree.</li> <li>- <code>Left</code> or <code>Indent</code>. A point to the left of the clicked item.</li> <li>- <code>Right</code>. A point to the right of the clicked item.</li> </ul> </li> <li>▶ During playback, Robot clicks in the center of the specified location.</li> <li>▶ <code>Position=%</code>. If <code>action%</code> is <code>VScrollTo</code> or <code>HScrollTo</code>, specifies the scroll bar value of the new scrolled-to position in the scroll box. Every scroll bar has an internal range, and this value is specific to that range.</li> </ul> |

Comments      When you act on a particular item in a tree object, Robot uses the text of the item (plus the text of any parent items) to identify it. In the following `recMethod$` value, the tree item labeled Service Division is a child of the tree item labeled Star Distribution Co.

```
"Name=tv_product;\;ItemText=Star Distribution Co->Service Division"
```

Note the two different parent/child separators — the *backslash* (\) separates the window object and its child object. The *pointer* (->) separates the parent text item from its child text item in the tree hierarchy.

## TreeView

When clicking on a branch that's very low in the tree hierarchy, or if branches have very long names, the maximum length for `recMethod$` strings might be exceeded. (The limit is 2,048 characters or less, depending on the circumstances.) If the limit is exceeded, Robot removes parent text items until the string length is within limits.

Here are some examples:

| <b>Tree Item Syntax</b> | <b>Meaning</b>                                                                                                                                                                                                             |
|-------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ItemA->ItemB            | This is a standard parent/child relationship used when the string limit has <i>not</i> been exceeded. It instructs Robot to look for ItemB in ItemA.                                                                       |
| ->ItemB                 | In this example, one or more parent items have been dropped. It instructs Robot to look for ItemB <i>anywhere</i> in the tree.                                                                                             |
| ->ItemB->>ItemC         | In this example, Robot looks for ItemC anywhere in the ItemB subhierarchy and <i>only</i> in the ItemB subhierarchy.                                                                                                       |
| .->ItemB                | This example introduces a new syntax element — the dot ( . ). A dot instructs Robot to look in the currently selected level of the tree. In this example, Robot looks for ItemB as a child of the currently selected item. |
| .->ItemB->>ItemC        | In this example, Robot looks for ItemC anywhere in the ItemB subhierarchy. ItemB is a child of the currently selected item.                                                                                                |

Using dot syntax is useful to avoid confusion when there are duplicate text items in the tree. It involves a sequence of at least two clicks — one to specify the current tree item, and one to specify some child (direct or indirect) of the current tree item.

### Example

This example clicks the expand button (+ sign) on the item identified by the text Employee Training. The item is in the first tree view control in the current context window (`ObjectIndex=1`). It is a child of the item Human Resources.

```
TreeView Click, "ObjectIndex=1;\;ItemText=Human Resources->Employee Training", "Location=Button"
```

### See Also

TreeViewVP

# TreeViewVP

Verification Point Command

»»SQA»

**Description** Establishes a verification point for a tree view control.

**Syntax** `Result = TreeViewVP (action%, recMethod$, parameters$)`

| Syntax Element           | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>action%</code>     | <p>The type of verification to perform. Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>CompareData</code>. Captures the data of the object and compares it to a recorded baseline. <code>parameters\$ VP</code> and <code>Type</code> are required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li> <li>▶ <code>CompareNumeric</code>. Captures the numeric value of the text of the object and compares it to the value of <code>parameters\$ Value</code> or <code>Range</code>. <code>parameters\$ VP</code> and either <code>Value</code> or <code>Range</code> are required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li> <li>▶ <code>CompareProperties</code>. Captures object properties information for the object and compares it to a recorded baseline. <code>parameters\$ VP</code> is required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li> <li>▶ <code>CompareText</code>. Captures the text of the object and compares it to a recorded baseline. <code>parameters\$ VP</code> and <code>Type</code> are required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li> </ul> |
| <code>recMethod\$</code> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>ID=%</code>. The object's internal Windows ID.</li> <li>▶ <code>ItemIndex=%</code>. The index of the tree view item acted upon. Used only after one of these parent values: <code>ID=%</code>, <code>ObjectIndex=%</code>, <code>Name=\$</code>, <code>Text=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>;\</code>).</li> <li>▶ <code>ItemText=\$</code>. The text of the tree view item acted upon. Used only after one of these parent values: <code>ID=%</code>, <code>ObjectIndex=%</code>, <code>Name=\$</code>, <code>Text=\$</code>. Parent/child values are separated by a backslash and semicolons (<code>;\</code>).</li> <li>▶ <code>Name=\$</code>. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be <code>Command1</code>.</li> <li>▶ <code>ObjectIndex=%</code>. The number of the object among all objects of the same type in the same window.</li> <li>▶ <code>Text=\$</code>. The text displayed on the object.</li> </ul>                         |

▶ ▶ ▶



| Syntax Element            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|---------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>parameters\$</code> | <p><b>Valid values:</b></p> <ul style="list-style-type: none"> <li>▶ <code>ExpectedResult=%</code>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> <li>- <code>PASS</code>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail.</li> <li>- <code>FAIL</code>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail.</li> </ul> </li> <li>▶ <code>Range=&amp;, &amp;</code>. Used with the action <code>CompareNumeric</code> when a numeric range comparison is being performed, as in <code>Range=2, 12</code> (test for numbers in this range). The values are inclusive.</li> <li>▶ <code>Type=\$</code>. Specifies the verification method to use for <code>CompareText</code> actions. The possible values are: <code>CaseSensitive</code>, <code>CaseInsensitive</code>, <code>FindSubStr</code>, <code>FindSubStrI</code> (case insensitive), and <code>UserDefined</code>. See <i>Comments</i> for more information. If <code>UserDefined</code> is specified, two additional parameters are required: <ul style="list-style-type: none"> <li>- <code>DLL=\$</code>. The full path and file name of the library that contains the function</li> <li>- <code>Function=\$</code>. The name of the custom function to use in comparing the text</li> </ul> </li> <li>▶ <code>Value=&amp;</code>. Used with the action <code>CompareNumeric</code> when a numeric equivalence comparison is being performed, as in <code>Value=25</code> (test against the value 25).</li> <li>▶ <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points.</li> <li>▶ <code>Wait=%, %</code>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10, 40</code> (retry the test every 10 seconds, but time out the test after 40 seconds).</li> </ul> |

**Comments** This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

With the `Type=$` parameter, `CaseSensitive` and `CaseInsensitive` require a full match between the current baseline text and the text captured during playback. With `FindSubStr` and `FindSubStrI`, the current baseline can be a substring of the text captured during playback. The substring can appear anywhere

in the playback text. To modify the current baseline text, double-click the verification point name in the Robot Asset pane (to the left of the script).

When you act on a particular item in a tree object, Robot uses the text of the item (plus the text of any parent items) to identify it. In the following *recMethod\$* value, the tree item labeled Service Division is a child of the tree item labeled Star Distribution Co.

```
"Name=tv_product;\;ItemText=Star Distribution Co->Service Division"
```

Note the two different parent/child separators — the *backslash* (\) separates the window object and its child object. The *pointer* (->) separates the parent text item from its child text item in the tree hierarchy.

When clicking on a branch that's very low in the tree hierarchy, or if branches have very long names, the maximum length for *recMethod\$* strings might be exceeded. (The limit is 2,048 characters or less, depending on the circumstances.) If the limit is exceeded, Robot removes parent text items until the string length is within limits. Here are some examples:

| Tree Item Syntax  | Meaning                                                                                                                                                                                                                  |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ItemA->ItemB      | This is a standard parent/child relationship used when the string limit has <i>not</i> been exceeded. It instructs Robot to look for ItemB in ItemA.                                                                     |
| ->ItemB           | In this example, one or more parent items have been dropped. It instructs Robot to look for ItemB <i>anywhere</i> in the tree.                                                                                           |
| ->ItemB->->ItemC  | In this example, Robot looks for ItemC anywhere in the ItemB subhierarchy and <i>only</i> in the ItemB subhierarchy.                                                                                                     |
| .->ItemB          | This example introduces a new syntax element — the dot (.). A dot instructs Robot to look in the currently selected level of the tree. In this example, Robot looks for ItemB as a child of the currently selected item. |
| .->ItemB->->ItemC | In this example, Robot looks for ItemC anywhere in the ItemB subhierarchy. ItemB is a child of the currently selected item.                                                                                              |

Using dot syntax is useful to avoid confusion when there are duplicate text items in the tree. It involves a sequence of at least two clicks — one to specify the current tree item, and one to specify some child (direct or indirect) of the current tree item.

**Example** This example captures the properties of the first tree view control in the window (ObjectIndex=1) and compares them to the recorded baseline in verification point TEST1A.

```
Result=TreeViewVP (CompareProperties, "ObjectIndex=1", "VP=TEST1A")
```

**See Also** TreeView

Trim

## Trim Function

---

**Description** Returns a copy of a string after removing all leading and trailing spaces.

**Syntax** **Trim**[\$](*expression*)

| Syntax Element    | Description                                                                                                                                                |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| \$                | Optional. If specified, the return type is <code>String</code> . If omitted, the function typically returns a Variant of <code>VarType 8 (String)</code> . |
| <i>expression</i> | The expression to trim. The expression can be a string, or it can be a numeric data type which Robot passes to the command as a string.                    |

**Comments** If the value of *string\$* is NULL, a Variant of `VarType 1 (Null)` is returned.

**Example** This example removes leading and trailing spaces from a string entered by the user.

```
Sub main
 Dim userstr as String
 userstr=InputBox("Enter a string with leading/trailing spaces")
 MsgBox "String is: " & Trim(userstr) & " with nothing after it."
End Sub
```

**See Also**

|          |               |
|----------|---------------|
| GetField | Mid function  |
| Left     | Mid statement |
| Len      | Right         |
| LTrim    | RTrim         |

## Type Statement

---

**Description** Declares a User-Defined data type.

**Syntax**

```
Type userType
 field1 As type1
 field2 As type2
 ...
End Type
```

| Syntax Element                | Description                                                                                                                                            |
|-------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>userType</i>               | The name of the user-defined type.                                                                                                                     |
| <i>field1</i> , <i>field2</i> | The name of a field in the user-defined type.                                                                                                          |
| <i>type1</i> , <i>type2</i>   | A data type: Integer, Long, Single, Double, Currency, String, String* <i>length</i> (for fixed-length strings), Variant, or another user-defined type. |

**Comments** The User-Defined data type declared by `Type` is then used in a `Dim` statement to declare a variable of that type. A user-defined type is sometimes referred to as a *record type* or a *structure type*.

*field* cannot be an array. However, arrays of user-defined types are allowed.

The `Type` statement is not valid inside of a procedure definition. To access the fields of a user-defined type, use this syntax:

```
TypeName.FieldName
```

To access the fields of an array of user-defined types, use this syntax:

```
ArrayName(index).FieldName
```

**Example** This example illustrates a `Type` and `Dim` statement. You must define a user-defined type before you can declare a variable of that type. The sub procedure then references a field within the user-defined type.

```
Type TestType ' name of User-Defined type
 Custno As Integer ' customer id field
 Custname As String ' customer name field
End Type

Sub main
 Dim MyType As TestType ' declare a variable for TestType
 Dim answer as String
 i: MyType.custname=InputBox("Enter a customer name:")
 If MyType.custname="" then
 Exit Sub
 End If
 answer=InputBox("Is the name: " & MyType.custname &
 " correct? (Y/N)")
 If answer="Y" or answer="y" then
 MsgBox "Thank you."
 Else
 MsgBox "Try again."
 Goto i
 End If
End Sub
```

**See Also** `DefType`  
`Dim`

Typeof

## Typeof Function

---

**Description** Returns a value indicating whether an object is of a given class (-1=TRUE, 0=FALSE).

**Syntax** If **Typeof** *objectVariable* Is *className* then. . .

| Syntax Element        | Description                         |
|-----------------------|-------------------------------------|
| <i>objectVariable</i> | The object to test.                 |
| <i>className</i>      | The class to compare the object to. |

**Comments** `Typeof` can only be used in an `If` statement and cannot be combined with other Boolean operators. That is, `Typeof` can only be used exactly as shown in the syntax above.

To test if an object does *not* belong to a class, use the following code structure:

```
If Typeof objectVariable Is className Then
 Rem Perform some action
Else
 Rem Perform some other action.
End If
```

**Example** None.

**See Also**

|              |              |
|--------------|--------------|
| CreateObject | Nothing      |
| GetObject    | Object Class |
| Is           | Class List   |
| New          |              |

## TypingDelays

Timing and Coordination Command

»»SQ«

**Description** Sets one or more keystroke delays during playback of the next `InputKeys` command.

**Syntax** **TypingDelays** *delayString\$*

| Syntax Element       | Description                                                                                                                                                                       |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>delayString\$</i> | A string containing one or more integers separated by commas. Each integer represents a delay time in milliseconds between keystrokes in the next <code>InputKeys</code> command. |



**Comments** Robot records this command if Record Think Time is selected in the General tab of the GUI Record Options dialog box. During playback, Robot performs the keystroke delays only if Use recorded typing delays is selected in the Playback tab of the GUI Playback Options dialog box.

The first integer value in *delayString\$* is always 0. Successive integers in *delayString\$* represent the delay time between keystrokes for the corresponding characters in the `InputKeys` command that follows the `TypingDelays` command.

Each `TypingDelays` command is preceded by a `SetThinkAvg` command, which sets the GUI think time between user actions.

Typing delays duration is the same in both Robot and LoadTest.

**Example** This example sets the typing delay between the keystrokes of the text "My Text" in the following `InputKeys` command.

```
SetThinkAvg 1500
TypingDelays "0, 160, 150, 650, 270, 270, 190"
InputKeys "My Text"
```

**See Also** `SetThinkAvg`  
`InputKeys`

## UBound

### Function

---

**Description** Returns the upper bound of the subscript range for the specified array.

**Syntax** `UBound(arrayname [, dimension ])`

| Syntax Element   | Description                   |
|------------------|-------------------------------|
| <i>arrayname</i> | The name of the array to use. |
| <i>dimension</i> | The dimension to use.         |

**Comments** The dimensions of an array are numbered starting with 1. If the *dimension* is not specified, 1 is used as a default.

`LBound` can be used with `UBound` to determine the length of an array.

## UBound

### Example

This example resizes an array if the user enters more data than can fit in the array. It uses LBound and UBound to determine the existing size of the array and ReDim to resize it. Option Base sets the default lower bound of the array to 1.

```
Option Base 1

Sub main
 Dim arrayvar() as Integer
 Dim count as Integer
 Dim answer as String
 Dim x, y as Integer
 Dim total
 total=0
 x=1
 count=InputBox("How many test scores do you have?")
 ReDim arrayvar(count)

start:

 Do until x=count+1
 arrayvar(x)=InputBox("Enter test score # " &x & ":")
 x=x+1
 Loop

 answer=InputBox$("Do you have more scores? (Y/N)")

 If answer="Y" or answer="y" then
 count=InputBox("How many more do you have?")
 If count<>0 then
 count=count+(x-1)
 ReDim Preserve arrayvar(count)
 Goto start
 End If
 End If

 x=LBound(arrayvar,1)
 count=UBound(arrayvar,1)

 For y=x to count
 total=total+arrayvar(y)
 Next y

 MsgBox "The average of the " & count & " scores is:
 " & Int(total/count)

End Sub
```

### See Also

|        |             |
|--------|-------------|
| Dim    | Option Base |
| Global | ReDim       |
| LBound | Static      |

## UCase

### Function

---

**Description** Returns a copy of a string after converting all lowercase letters to uppercase.

**Syntax** **UCase** [*\$*] (*string\$*)

| Syntax Element  | Description                                                                                                                                            |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>\$</i>       | Optional. If specified, the return type is <i>String</i> . If omitted, the function typically returns a Variant of <i>VarType 8</i> ( <i>String</i> ). |
| <i>string\$</i> | An expression that evaluates to a string.                                                                                                              |

**Comments** The translation is based on the country specified in the Windows Control Panel. *UCase* accepts expressions of type *string*. *UCase* accepts any type of argument and will convert the input value to a string. If the value of *string\$* is *Null*, a Variant of *VarType 1* (*Null*) is returned.

**Example** This example converts a file name entered by a user to all uppercase letters.

```
Option Base 1
Sub main
 Dim filename as String
 filename=InputBox("Enter a filename: ")
 filename=UCase(filename)
 MsgBox "The filename in uppercase is: " & filename
End Sub
```

**See Also** *Asc*      *LCase*

## Unlock

### Statement

---

**Description** Restores access to an open file (releases the lock).

**Syntax** **Unlock** [#]*filename%* [, {*record&* | [*start&*] To *end&*}]

| Syntax Element     | Description                                               |
|--------------------|-----------------------------------------------------------|
| <i>filename%</i>   | An integer expression identifying the open file.          |
| <i>record&amp;</i> | Number of the starting record to unlock.                  |
| <i>start&amp;</i>  | Number of the first record or byte offset to lock/unlock. |
| <i>end&amp;</i>    | Number of the last record or byte offset to lock/unlock.  |

## Unlock

**Comments**     The *filenumber%* is the number used in the Open statement of the file.

For Binary mode, *start&*, and *end&* are byte offsets. For Random mode, *start&*, and *end&* are record numbers. If *start&* is specified without *end&*, only the record or byte at *start&* is locked. If To *end&* is specified without *start&*, all records or bytes from record number or offset 1 to *end&* are locked.

For Input, Output and Append modes, *start&*, and *end&* are ignored and the whole file is locked.

Lock and Unlock always occur in pairs with identical parameters. All locks on open files must be removed before closing the file or unpredictable results occur.

**Example**        This example locks a file that is shared by others on a network, if the file is already in use. The second sub procedure, CREATEFILE, creates the file used by the main sub procedure.

```
Declare Sub createfile
Sub main
 Dim btngrp, icongrp, defgrp, answer, msgabort
 Dim noaccess as Integer
 Dim msgstop as Integer
 Dim acctname as String
 noaccess=70
 msgstop=16
 Call createfile
 On Error Resume Next
 btngrp=1
 icongrp=64
 defgrp=0
 answer=MsgBox("Open the account file?" & Chr(10),
 btngrp + icongrp + defgrp)
 If answer=1 then
 Open "C:\TEMP001" for Input as #1
 If Err=noaccess then
 msgabort=MsgBox("File Locked",msgstop,"Aborted")
 Else
 Lock #1
 Line Input #1, acctname
 MsgBox "The first account name is: " & acctname
 Unlock #1
 End If
 Close #1
 End If
 Kill "C:\TEMP001"
End Sub

Sub createfile()
 Rem Put the letters A-J into the file
 Dim x as Integer
 Open "C:\TEMP001" for Output as #1
 For x=1 to 10
 Write #1, Chr(x+64)
 Next x
 Close #1
End Sub
```

**See Also**     Lock     Open

## UserDefinedTC

Verification Point Command

»»SQA»

This command is obsolete and should not be used. It continues to be supported to maintain the upward compatibility of your existing scripts.

## Val

Function

**Description** Returns the numeric value of the first number found in the specified string.

**Syntax** **Val**(*string\$*)

| Syntax Element  | Description                              |
|-----------------|------------------------------------------|
| <i>string\$</i> | A string expression containing a number. |

**Comments** If no number is found, Val returns 0.

Val ignores spaces anywhere in the source string. Val also ignores non-numeric characters that appear after the number. If non-numeric characters appear before the number, Val returns 0.

**Example** This example tests the value of the variable profit and displays 0 for profit if it is a negative number. The sub procedure uses Sgn to determine whether profit is positive, negative or zero.

```
Sub main
 Dim profit as Single
 Dim expenses
 Dim sales
 expenses=InputBox("Enter total expenses: ")
 sales=InputBox("Enter total sales: ")
 profit=Val(sales)-Val(expenses)
 If Sgn(profit)=1 then
 MsgBox "Yeah! We turned a profit!"
 ElseIf Sgn(profit)=0 then
 MsgBox "Okay. We broke even."
 Else
 MsgBox "Uh, oh. We lost money."
 End If
End Sub
```

**See Also**

|      |        |        |
|------|--------|--------|
| CCur | CSng   | Format |
| Cdbl | CStr   | Str    |
| CInt | CVar   |        |
| CLng | CVDate |        |

VarType

## VarType Function

---

**Description** Returns the Variant type of the specified Variant variable (0-9).

**Syntax** **VarType**(*varname*)

| Syntax Element | Description                  |
|----------------|------------------------------|
| <i>varname</i> | The Variant variable to use. |

**Comments** The value returned by VarType is one of the following:

| Ordinal | Representation |
|---------|----------------|
| 0       | (Empty)        |
| 1       | Null           |
| 2       | Integer        |
| 3       | Long           |
| 4       | Single         |
| 5       | Double         |
| 6       | Currency       |
| 7       | Date           |
| 8       | String         |
| 9       | Object         |

**Example** This example returns the type of a variant.

```
Sub main
 Dim x
 Dim myarray(8)
 Dim retval
 Dim retstr
 myarray(1)=Null
 myarray(2)=0
 myarray(3)=39000
 myarray(4)=CSng(10^20)
 myarray(5)=10^300
 myarray(6)=CCur(10.25)
 myarray(7)=Now
 myarray(8)="Five"
```

```

For x=0 to 8
 retval=Vartype(myarray(x))
 Select Case retval
 Case 0
 retstr=" (Empty)"
 Case 1
 retstr=" (Null)"
 Case 2
 retstr=" (Integer)"
 Case 3
 retstr=" (Long)"
 Case 4
 retstr=" (Single)"
 Case 5
 retstr=" (Double)"
 Case 6
 retstr=" (Currency)"
 Case 7
 retstr=" (Date)"
 Case 8
 retstr=" (String)"
 End Select
 If retval=1 then
 myarray(x)="[null]"
 ElseIf retval=0 then
 myarray(x)="[empty]"
 End If
 MsgBox "The variant type for " &myarray(x) & " is:
 " &retval &retstr
Next x
End Sub

```

See Also      [IsDate](#)            [IsNull](#)  
                  [IsEmpty](#)           [IsNumeric](#)

## WebSiteVP

### Verification Point Command



**Description**      Tests for defects (such as missing or broken links) on a Web site, or compares Web sites.

**Syntax**            *Result* = **WebSiteVP**(*action%*, " ", *parameters\$*)

| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                                         |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i> | The type of verification to perform. Valid values: <ul style="list-style-type: none"> <li>▶ SiteCheck. Scans for defects on a single Web site.</li> <li>▶ Compare. Compares two Web sites. These can be the same Web site at two different periods of time, mirror sites, or different Web sites. <i>parameters\$</i> VP is required; <i>ExpectedResult</i> is optional.</li> </ul> |



## WebSiteVP

▶ ▶ ▶

| Syntax Element       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| " "                  | The second argument is always left blank.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <i>parameters</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>ExpectedResult=%</code>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> <li>– <code>PASS</code>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail.</li> <li>– <code>FAIL</code>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail.</li> </ul> </li> <li>▶ <code>VP=\$</code>. The verification point ID. IDs must be unique within a script. Required for all verification points.</li> </ul> |

### Comments

This function returns 1 if the verification point passes or 0 if the verification point fails.

When *action* is set to `SiteCheck`, `WebSiteVP` tests for the type of defects that you specify in the Rational SiteCheck Scan Options dialog box. You specify the types of defects to test for when you select the Web Site Scan verification point during recording (click Insert → Verification Point → Web Site Scan).

You can use the `SiteCheck` setting to save the current version of a site in a site map. Then, you can perform a `WebSiteVP Compare` and use this site map as a baseline to compare against a later version of the site.

When *action* is set to `Compare`, `WebSiteVP` compares a baseline site and a comparison site. For example, you can compare a previously saved site map with the current version of a site, compare mirror sites, or compare any two sites. The comparison is based on files that have been added, modified, or deleted since the baseline scan. The comparison involves these areas:

- ▶ HTML files
- ▶ Image files
- ▶ Orphan files
- ▶ External links
- ▶ Other files and links

A `WebSiteVP` verification point passes if no defects or differences are found in any of the areas you specify on the Scan Options dialog box. If one or more



defects or differences are found in any of the areas you specify, the verification point fails.

Both PASS and FAIL results are reported in the LogViewer. For an explanation of any failure, double-click on the entry in the LogViewer.

For information about SiteCheck, see the Rational SiteCheck Help.

**Example** This example establishes the Web site verification point CKLINKSA.

```
Result = WebsiteVP(SiteCheck, "", "VP=CKLINKSA")
```

**See Also** None.

## Weekday Function

---

**Description** Returns the day of the week for the specified date-time value.

**Syntax** `weekday(date)`

| Syntax Element | Description                                 |
|----------------|---------------------------------------------|
| <i>date</i>    | An expression containing a date time value. |

**Comments** The Weekday function returns an integer between 1 and 7, inclusive (1=Sunday, 7=Saturday).

Weekday accepts any expression, including strings, and attempts to convert the input value to a date value.

The return value is a Variant of VarType 2 (Integer). If the value of *date* is NULL, a Variant of VarType 1 (Null) is returned.

**Example** This example finds the day of the week on which New Year's Day will fall in the year 2000.

```
Sub main
 Dim newyearsday, daynumber, msgtext
 Dim newday as Variant
 Const newyear=2000
 Const newmonth=1
 Let newday=1
 newyearsday=DateSerial(newyear, newmonth, newday)
 daynumber=Weekday(newyearsday)
 msgtext="New Year's day 2000 is a " & Format(daynumber, "dddd")
 MsgBox msgtext
End Sub
```

## While...Wend

|          |                |        |        |
|----------|----------------|--------|--------|
| See Also | Date function  | Hour   | Now    |
|          | Date statement | Minute | Second |
|          | Day            | Month  | Year   |

## While...Wend

### Statement

---

**Description** Controls a repetitive action.

**Syntax** **While** *condition*  
*statement\_block*  
**Wend**

| Syntax Element         | Description                                                      |
|------------------------|------------------------------------------------------------------|
| <i>condition</i>       | An expression that evaluates to TRUE (non-zero) or FALSE (zero). |
| <i>statement_block</i> | A series of statements to execute if <i>condition</i> is TRUE.   |

**Comments** The *statement\_block* statements are until *condition* becomes 0 (FALSE). The **While** statement is included in SQABasic for compatibility with older versions of Basic. The **Do** statement is a more general and powerful flow control statement.

**Example** This example opens a series of customer files and checks for the string \*Overdue\* in each file. It uses **While...Wend** to loop through the C:\TEMP00? files. These files are created by the sub procedure CREATEFILES.

```
Declare Sub createfiles
Sub main
 Dim custfile as String
 Dim aline as String
 Dim pattern as String
 Dim count as Integer

 Call createfiles
 Chdir "C:\"
 custfile=Dir$("TEMP00?")
 pattern="*" + "Overdue" + "**"
```

```

While custfile <> ""
 Open custfile for input as #1
 On Error goto atEOF
 Do
 Line Input #1, aline
 If aline Like pattern Then
 count=count+1
 End If
 Loop
nxtfile:
 On Error GoTo 0
 Close #1
 custfile = Dir$
Wend

 If count<>0 then
 MsgBox "Number of overdue accounts: " & count
 Else
 MsgBox "No accounts overdue"
 End If

 Kill "C:\TEMP001"
 Kill "C:\TEMP002"
 Exit Sub

atEOF:
 Resume nxtfile

End Sub

Sub createfiles()

 Dim odue as String
 Dim ontime as String
 Dim x

 Open "C:\TEMP001" for OUTPUT as #1
 odue="*" + "Overdue" + "*"
 ontime="*" + "On-Time" + "*"

 For x=1 to 3
 Write #1, odue
 Next x

 For x=4 to 6
 Write #1, ontime
 Next x

 Close #1
 Open "C:\TEMP002" for Output as #1
 Write #1, odue
 Close #1

End Sub

```

See Also      Do...Loop

Width

## Width

Statement

---

**Description** Sets the output line width for an open file.

**Syntax** `width [#]filename%, width%`

| Syntax Element   | Description                                                 |
|------------------|-------------------------------------------------------------|
| <i>filename%</i> | An integer expression for the open file to use.             |
| <i>width%</i>    | An integer expression for the width of the line (0 to 255). |

**Comments** *Filename%* is the number assigned to the file when it is opened. See the [Open statement](#) for more information.

A value of zero (0) for *width%* indicates there is no line length limit. The default *width%* for a file is zero (0).

**Example** This example puts five spaces and the string ABCD to a file. The five spaces are derived by taking 15 MOD 10, or the remainder of dividing 15 by 10.

```
Sub main
 Dim str1 as String
 Dim x as String*10

 str1="ABCD"
 Open "C:\TEMP001" For Output As #1
 width #1, 10
 Print #1, Spc(15); str1
 Close #1
 Open "C:\TEMP001" as #1 Len=12
 Get #1, 1,x
 MsgBox "The contents of the file is: " & x
 Close #1
 Kill "C:\TEMP001"
End Sub
```

**See Also** [Open](#)  
[Print](#)

# Window

User Action Command

▶▶SQA▶

Description Performs an action on a window.

Syntax **Window** *action%*, *recMethod\$*, *parameters\$*

| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>action%</i> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <b>CloseWin</b>. Closes the specified window. <i>parameters\$</i> is left blank for this action, as in:<br/>           Window CloseWin, "Caption=App1", ""</li> <li>▶ <b>MouseClicked</b>. The clicking of the left, center, or right mouse button, either alone or in combination with one or more shifting keys (Ctrl, Alt, Shift). When <i>action%</i> contains a mouse-click value, <i>parameters\$</i> must contain <i>Coords=x,y</i>.</li> <li>▶ <b>MouseDown</b>. The dragging of the mouse while mouse buttons and/or shifting keys (Ctrl, Alt, Shift) are pressed. When <i>action%</i> contains a mouse-drag value, <i>parameters\$</i> must contain <i>Coords=x1,y1,x2,y2</i>. See Appendix E for a list of mouse click and drag values.</li> <li>▶ <b>MoveTo</b>. A repositioning action for which the <i>x,y</i> coordinates specify the position of the top left corner to which the window is to be moved, relative to its parent window, as in:<br/>           Window MoveTo, "Caption=Mortgage-Prequalifier", "Coords=99,109"</li> <li>▶ <b>OpenIcon</b>. Opens an iconized window. <i>parameters\$</i> is left blank for this action, as in:<br/>           Window OpenIcon, "Caption=App1", ""</li> <li>▶ <b>ResetTestContext</b>. Restores the test context to be the context window. In other words, the test context is set back to its state prior to the last <b>SetTestContext</b> action.</li> <li>▶ <b>Resize</b>. Resizes the specified window, based on its top left and bottom right coordinates (<i>x1,y1,x2,y2</i>) as in:<br/>           Window Resize, "Caption=Program Manager", "Coords=5,2,100,80"</li> </ul> |

▶ ▶ ▶



| Syntax Element  | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                 |                |             |                |                |              |            |              |           |           |
|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|----------------|-------------|----------------|----------------|--------------|------------|--------------|-----------|-----------|
|                 | <ul style="list-style-type: none"> <li>▶ <b>RestorePos.</b> Restores the specified window to its original size and position. <i>parameters\$</i> is left blank for this action, as in: <ul style="list-style-type: none"> <li>▶ Window RestorePos, "Caption=App1", ""</li> <li>▶ <i>ScrollAction.</i> One of these scroll actions: <table border="0" style="margin-left: 40px;"> <tr> <td>ScrollPageRight</td> <td>ScrollPageDown</td> </tr> <tr> <td>ScrollRight</td> <td>ScrollLineDown</td> </tr> <tr> <td>ScrollPageLeft</td> <td>ScrollPageUp</td> </tr> <tr> <td>ScrollLeft</td> <td>ScrollLineUp</td> </tr> <tr> <td>HScrollTo</td> <td>VScrollTo</td> </tr> </table> <p>HScrollTo and VScrollTo take the required parameter <code>Position=%</code>.</p> <p>If Robot cannot interpret the action being applied to a scroll bar, which happens with certain custom standalone scroll bars, it records the action as a click or drag.</p> </li> </ul> </li> <li>▶ <b>SetContext.</b> Establishes the context window for all Object commands that follow. <p>During playback, Robot locates the specified window. If the window is the active window, it remains active. If the window is not the active window, Robot takes one of these actions:</p> <ul style="list-style-type: none"> <li>- If <i>parameters\$</i> is an empty string (" ") or contains <code>Activate=1</code>, Robot makes the window the active window.</li> <li>- If <i>parameters\$</i> contains <code>Activate=0</code>, Robot will not make the window the active window.</li> <li>- If <i>recMethod\$</i> does not contain the <code>State=Disabled</code> qualifier, Robot makes the window the active window.</li> </ul> <p>Setting the context window defines an internal state for Robot. If this command fails during playback (for example, if the specified window cannot be found), an error is logged, but playback continues regardless of the playback option for script command failures.</p> <p>The context for all Object commands that follow is assumed to be the current context window. For example, when Robot plays back a Command button command, it assumes the button is in the current context window.</p> </li> </ul> | ScrollPageRight | ScrollPageDown | ScrollRight | ScrollLineDown | ScrollPageLeft | ScrollPageUp | ScrollLeft | ScrollLineUp | HScrollTo | VScrollTo |
| ScrollPageRight | ScrollPageDown                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                 |                |             |                |                |              |            |              |           |           |
| ScrollRight     | ScrollLineDown                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                 |                |             |                |                |              |            |              |           |           |
| ScrollPageLeft  | ScrollPageUp                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                 |                |             |                |                |              |            |              |           |           |
| ScrollLeft      | ScrollLineUp                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                 |                |             |                |                |              |            |              |           |           |
| HScrollTo       | VScrollTo                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                 |                |             |                |                |              |            |              |           |           |



▶ ▶ ▶

| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                | <ul style="list-style-type: none"> <li>▶ <b>SetPosition.</b> Sets the size, position, and state of a window. The size and position are specified by top left and bottom right coordinates: <i>x1,y1,x2,y2</i> (relative to the parent window or Desktop). The state is specified by one of the following keywords: MINIMIZED, MAXIMIZE, or NORMAL. For example: <pre style="margin-left: 2em;">Window SetPosition, "Caption=File",     "Coords=5,2,100,80;Status=NORMAL"</pre> </li> <li>▶ <b>SetTestContext.</b> Establishes the test context for subsequent verification point commands. It has no effect on standard Object commands. <i>parameters\$</i> is left blank for this action, as in: <pre style="margin-left: 2em;">Window SetTestContext,     "Caption=Classics Online", ""</pre> <p>By default, the test context is the same as the context window as set by the <code>SetContext</code> action.</p> <p><code>SetTestContext</code> is used when you need to insert a verification point for an object or window that is outside of the current context window (for example, if you want to test the properties of a button in one dialog box while acting on a different dialog box).</p> <p><b>Note:</b> The <code>SetContext</code> action sets both the context window and the test context. In other words, <code>SetContext</code> overrides any prior <code>SetTestContext</code> action.</p> </li> <li>▶ <b>WMaximize.</b> Maximizes the specified window. <i>parameters\$</i> is left blank for this action, as in: <pre style="margin-left: 2em;">Window WMaximize,     "Caption=Classics Online", ""</pre> </li> <li>▶ <b>WMinimize.</b> Minimizes the specified window. <i>parameters\$</i> is left blank for this action, as in: <pre style="margin-left: 2em;">Window WMinimize,     "Caption=Classics Online", ""</pre> </li> </ul> |

▶ ▶ ▶

▶ ▶ ▶

| Syntax Element           | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>recMethod\$</code> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ [empty quotes]. If the recognition method is empty, Robot performs the action on the current context window, as specified by the last <code>SetContext</code> action. For example, the following commands minimize the window identified by the caption <code>App1</code>: <pre>Window SetContext, "Caption=App1", "" Window WMinimize, "", ""</pre> </li> <li>▶ <code>Caption=\$</code>. The text that appears in the window's title bar. 512 characters maximum. The wildcards <code>?</code> and <code>*</code> are supported. (See <i>Establishing Context through a Window Command</i> in Chapter 4 for information.)</li> <li>▶ <code>ChildWindow</code>. Indicates that the window specified by the recognition method is a child of the current context window. It is only used in conjunction with another method. This qualifier is necessary when setting the context or acting upon windows that are children of other windows. <p>The following example minimizes the window <code>Book1</code> in Microsoft Excel:</p> <pre>Window SetContext,     "Caption=Microsoft Excel", "" Window WMinimize,     "Caption=Book1;ChildWindow", ""</pre> </li> <li>▶ <code>Class=\$</code>. The window's class name.</li> <li>▶ <code>CurrentWindow</code>. Sets the context to the currently-active window. Used only programmatically, as in: <pre>Window SetContext, "CurrentWindow", ""</pre> <p>This recognition method is useful when you want to set the context or act upon the window that is currently active, even though that may not be the same window each time the command is played back. This recognition method should not be used in conjunction with any other recognition methods.</p> </li> <li>▶ <code>Level=%</code>. <code>Level</code> is combined with another recognition method when the other method does not uniquely identify the windows. For example, if there are multiple windows with the same caption, and <code>Caption</code> is the recognition method being used. The <code>Level</code> qualifier tells Robot which one of the similarly-identified windows should be targeted for the action, based on the Windows' Z-Order.</li> </ul> |

▶ ▶ ▶



▶ ▶ ▶

| Syntax Element      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                     | <p>The first window is assigned <code>Level=1</code>, the second <code>Level=2</code>, and so on. <code>Level</code> serves as a clarifier only and is used only after all other methods have been attempted.</p> <ul style="list-style-type: none"> <li>▶ <code>Name=\$</code>. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be <code>Command1</code>.</li> <li>▶ <code>State=\$</code>. An optional qualifier for any other recognition method. There are two possible values for this setting: <code>Enabled</code> and <code>Disabled</code>. When Robot looks for a specified window, it checks the state of that window against an expected value (<code>State=Enabled</code> is the default). Robot only records this setting if the object is disabled.</li> <li>▶ <code>VisualText=\$</code>. An optional setting used to identify an object by its visible text. It is for user clarification only and does not affect object recognition.</li> <li>▶ <code>WindowTag=\$</code>. An optional setting used during Web testing to identify a particular instance of the browser. If a browser ID exists, it is defined in the <code>StartBrowser</code> command.</li> </ul> |
| <i>parameters\$</i> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ If <i>action%</i> is <code>SetContext</code>, <i>parameters\$</i> contains one of these values: <ul style="list-style-type: none"> <li>- <code>Activate=1</code>. Robot makes the window the active window.<br/>This is the default setting. If <i>parameters\$</i> is an empty string (" "), Robot makes the window the active window.</li> <li>- <code>Activate=0</code>. Robot does not make the window the active window.</li> </ul> </li> <li>▶ <b>Note:</b> During recording, Robot generates a <i>parameters\$</i> value of either " " if the window is to be made the active window, or <code>Activate=0</code> if the window is not to be made the active window.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |

▶ ▶ ▶

## Window

▶ ▶ ▶

| Syntax Element | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                | <ul style="list-style-type: none"> <li>▶ <code>Coords=x,y</code>. If <code>action%</code> is a mouse click, specifies the <code>x,y</code> coordinates of the click, relative to the top left of the object.<br/>If <code>action%</code> is <code>MoveTo</code>, specifies the <code>x,y</code> coordinates to which the window object is to be moved. The coordinates are relative to the top left of the parent window or Desktop, if there is no parent.</li> <li>▶ <code>Coords=x1,y1,x2,y2</code>. If <code>action%</code> is a mouse drag, specifies the coordinates, where <code>x1,y1</code> are the starting coordinates of the drag, and <code>x2,y2</code> are the ending coordinates. The coordinates are relative to the top left of the object.<br/>If <code>action%</code> is <code>Resize</code> or <code>SetPosition</code>, the coordinates correspond to the top left and bottom right coordinates of the resized window. If <code>action%</code> is <code>SetPosition</code>, the "Status=" parameter is also used.</li> <li>▶ <code>Position=%</code>. If <code>action%</code> is <code>VScrollTo</code> or <code>HScrollTo</code>, specifies the scroll bar value of the new position of the scroll box. Every scroll bar has an internal range and this parameter value is specific to that range.</li> <li>▶ <code>Status=\$</code>. If <code>action%</code> is <code>SetPosition</code>, specifies the state of the window: <code>NORMAL</code>, <code>MINIMIZED</code> or <code>MAXIMIZED</code>.</li> </ul> |

**Comments** In this document, a *window* is a top-level object on the desktop. For example, a dialog box is typically a top-level desktop object.

**Example** This example double-clicks the window identified by the caption `International` at the `x,y` coordinates of `184,15`.

```
Window DblClick, "Caption=International", "Coords=184,15"
```

**See Also**

|                           |                        |
|---------------------------|------------------------|
| <code>ComboListBox</code> | <code>ListBox</code>   |
| <code>EditBox</code>      | <code>ScrollBar</code> |

# WindowVP

## Verification Point Command

»»SQA»

**Description** Establishes a verification point for a window.

**Syntax** `Result = WindowVP (action%, recMethod$, parameters$)`

| Syntax Element       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>action%</code> | <p>The type of verification to perform. Valid values:</p> <ul style="list-style-type: none"> <li>▶ <code>CompareDataWindow</code>. Captures the data stored in the PowerBuilder DataWindow and compares it to a recorded baseline. <code>parameters\$ VP</code> is required; <code>ExpectedResult</code> and <code>Wait</code> are optional.<br/><b>Note:</b> This action is only used when the DataWindow object is a Window itself.</li> <li>▶ <code>CompareImage</code>. Captures a bitmap image of the specified window and compares it to a recorded baseline. <code>parameters\$ VP</code> is required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li> <li>▶ <code>CompareMenu</code>. Captures the specified window's menu information and compares it to a recorded baseline. <code>parameters\$ VP</code> is required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li> <li>▶ <code>CompareNumeric</code>. Captures the numeric value of the text of the object and compares it to the value of <code>parameters\$ Value</code> or <code>Range</code>. <code>parameters\$ VP</code> and either <code>Value</code> or <code>Range</code> are required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li> <li>▶ <code>CompareProperties</code>. Captures the object properties information of the Window and all of its children, and compares this to the recorded baseline. <code>parameters\$ VP</code> is required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li> <li>▶ <code>CompareText</code>. Captures the text in the title bar of a specified window and compares it to a recorded baseline. <code>parameters\$ VP</code> and <code>Type</code> are required; <code>ExpectedResult</code> and <code>Wait</code> are optional.</li> <li>▶ <code>DoesNotExist</code>. Checks whether a specified window no longer exists at playback. <code>parameters\$ VP</code> is required; <code>ExpectedResult</code>, <code>Status</code>, and <code>Wait</code> are optional.<br/><b>Note:</b> This action cannot be accessed during recording. It must be inserted manually.</li> <li>▶ <code>Exists</code>. Checks whether a specified window exists at playback. <code>parameters\$ VP</code> is required; <code>ExpectedResult</code>, <code>Status</code>, and <code>Wait</code> are optional.</li> </ul> |

▶ ▶ ▶

▶ ▶ ▶

| Syntax Element           | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|--------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>recMethod\$</code> | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ [empty quotes]. If the recognition method is empty, Robot performs the action on the current test context window, as specified by the last <code>SetContext</code> or <code>SetTestContext</code> action.</li> <li>▶ <code>Caption=\$</code>. The text that appears in the window's title bar. 512 characters maximum. The wildcards <code>?</code> and <code>*</code> are supported. (See <i>Establishing Context through a Window Command</i> in Chapter 4 for information.)</li> <li>▶ <code>ChildWindow</code>. Indicates that the window specified by the recognition method is a child of the current context window. It is only used in conjunction with another method. This qualifier is necessary when acting upon windows that are children of other windows.</li> <li>▶ <code>Class=\$</code>. The window's class name.</li> <li>▶ <code>CurrentWindow</code>. Specifies the windows that is currently active. This recognition method is useful when you want to act upon the active window, even though that may not be the same window each time the command is played back. This recognition method should not be used in conjunction with any other methods.</li> <li>▶ <code>Level=%</code>. <code>Level</code> is combined with another recognition method when the other recognition method does not uniquely identify the windows. For example, if there are multiple windows with the same caption, and <code>Caption</code> is the recognition method being used. The <code>Level</code> qualifier tells Robot which one of the similarly-identified windows should be targeted for the action, based on the Windows' Z-Order. The first window is assigned "<code>Level=1</code>", the second "<code>Level=2</code>", and so on. <code>Level</code> serves as a clarifier only and is used only after all other methods have been attempted.</li> <li>▶ <code>Name=\$</code>. A name that a developer assigns to an object to uniquely identify the object in the development environment. For example, the object name for a command button might be <code>Command1</code>.</li> </ul> |

▶ ▶ ▶



| Syntax Element       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>parameters</i> \$ | <p>Valid values:</p> <ul style="list-style-type: none"> <li>▶ <b>ExpectedResult=%</b>. Specifies whether you expect this verification point to pass (baseline result matches playback result) or fail (baseline result does not match playback result). Valid values: <ul style="list-style-type: none"> <li>- <b>PASS</b>. The default. If the baseline and playback results match as expected, the LogViewer reports Pass. If they do not match, the LogViewer reports Fail.</li> <li>- <b>FAIL</b>. If the baseline and playback results do not match as expected, the LogViewer reports Pass. If they do match, the LogViewer reports Fail.</li> </ul> </li> <li>▶ <b>Range=&amp; , &amp;</b>. Used with the action <code>CompareNumeric</code> when a numeric range comparison is being performed, as in <code>Range=2 , 12</code> (test for numbers in this range). The values are inclusive.</li> <li>▶ <b>Status=\$</b>. An optional parameter used with the <b>Exists</b> action. When used, the status of the window is also verified. The possible values for this parameter are: <code>NORMAL</code>, <code>MINIMIZED</code>, and <code>MAXIMIZED</code></li> <li>▶ <b>Type=\$</b>. Specifies the verification method to use for <code>CompareText</code> actions. The possible values are: <code>CaseSensitive</code>, <code>CaseInsensitive</code>, <code>FindSubStr</code>, <code>FindSubStrI</code> (case insensitive), and <code>UserDefined</code>. See <i>Comments</i> for more information. If <code>UserDefined</code> is specified, two additional parameters are required: <ul style="list-style-type: none"> <li>- <b>DLL=\$</b>. The full path and file name of the library that contains the function</li> <li>- <b>Function=\$</b>. The name of the custom function to use in comparing the text. For example: <pre style="margin-left: 40px;">Result = WindowVP (CompareText ,     "Class=MyWndClass" ,     "VP=UDTEXT ;Type=UserDefined ;     DLL=C:\MYFUNC.DLL ;     Function=VerifyLength" )</pre> </li> </ul> </li> <li>▶ <b>Value=&amp;</b>. Used with the action <code>CompareNumeric</code> when a numeric equivalence comparison is being performed, as in <code>Value=25</code> (test against the value 25).</li> <li>▶ <b>VP=\$</b>. The verification point ID. IDs must be unique within a script. Required for all verification points.</li> <li>▶ <b>Wait=% , %</b>. A Wait State that specifies the verification point's Retry value and a Timeout value, as in <code>Wait=10 , 40</code> (retry the test every 10 seconds, but time out the test after 40 seconds).</li> </ul> |

## With

**Comments** This function returns 1 if the action performed passes or 0 if the action performed fails. See the LogViewer for an explanation of any failures.

In this document, a *window* is a top-level object on the desktop. For example, a dialog box is typically a top-level desktop object.

With the `Type=$` parameter, `CaseSensitive` and `CaseInsensitive` require a full match between the current baseline text and the text captured during playback. With `FindSubStr` and `FindSubStrI`, the current baseline can be a substring of the text captured during playback. The substring can appear anywhere in the playback text. To modify the current baseline text, double-click the verification point name in the Robot Asset pane (to the left of the script).

Verification points that check for a window's existence are not stored in the repository and do not appear in Robot's Asset pane.

**Example** This example captures a bitmap image of the window identified by the `Caption Paint` and compares it to a recorded baseline in verification point `PICT1A`.

```
Result = WindowVP (CompareImage, "Caption=Paint", "VP=PICT1A")
```

**See Also**

|                         |                          |
|-------------------------|--------------------------|
| <code>ComboBoxVP</code> | <code>ListBoxVP</code>   |
| <code>EditBoxVP</code>  | <code>ScrollBarVP</code> |

## With Statement

---

**Description** Executes a series of statements on a specified variable.

**Syntax**

```
With variable
 statement_block
End With
```

| Syntax Element         | Description                                                              |
|------------------------|--------------------------------------------------------------------------|
| <i>variable</i>        | The variable to be changed by the statements in <i>statement_block</i> . |
| <i>statement_block</i> | The statements to execute.                                               |

**Comments** *Variable* can be an Object data type or a user-defined data type. `With` statements can be nested.

**Example** This example creates a user-defined data type named `CustType`, declares an instance of the data type called `Customer`, then uses the `With` statement to fill in values for the fields in `Customer`.

```

Type CustType
 name as String
 ss as String
 salary as Single
 dob as Variant
 street as String
 apt as Variant
 city as String
 state as String
End Type

Sub main
 Dim Customer as CustType
 Dim msgtext
 With Customer
 .name="John Jones"
 .ss="037-67-2947"
 .salary=60000
 .dob=#10-09-65#
 .street="15 Chester St."
 .apt=28
 .city="Cambridge"
 .state="MA"
 End With
 msgtext=Chr(10) & "Name:" & Space(5) & Customer.name & Chr(10)
 msgtext=msgtext & "SS#: " & Space(6) & Customer.ss & Chr(10)
 msgtext=msgtext & "D.O.B:" & Space(4) & Customer.dob
 MsgBox "Done with: " & Chr(10) & msgtext
End Sub

```

**See Also** `Type...End Type`

## Write Statement

---

**Description** Writes data to an open sequential file.

**Syntax** `Write #filenumber% [,expressionlist]`

| Syntax Element        | Description                                     |
|-----------------------|-------------------------------------------------|
| <i>filenumber%</i>    | An integer expression for the open file to use. |
| <i>expressionlist</i> | One or more values to write to the file.        |

Write

**Comments**     The file must be opened in Output or Append mode. *filenumber%* is the number assigned to the file when it is opened. See the `Open` statement for more information.

If *expressionlist* is omitted, the `Write` statement writes a blank line to the file. (See `Input` for more information.)

**Example**       This example writes a variable to a disk file based on a comparison of its last saved time and the current time.

```
Sub main
 Dim tempfile
 Dim filetime, curtime
 Dim msgtext
 Dim acctno(100) as Single
 Dim x, I
 tempfile="C:\TEMP001"
 Open tempfile For Output As #1
 filetime=FileDateTime(tempfile)
 x=1
 I=1
 acctno(x)=0
 Do
 curtime=Time
 acctno(x)=InputBox("Enter an account number (99 to end):")
 If acctno(x)=99 then
 If x=1 then Exit Sub
 For I=1 to x-1
 Write #1, acctno(I)
 Next I
 Exit Do
 ElseIf (Minute(filetime)+2)<=Minute(curtime) then
 For I=I to x-1
 Write #1, acctno(I)
 Next I
 End If
 x=x+1
 Loop
 Close #1
 x=1
 msgtext="Contents of C:\TEMP001 is:" & Chr(10)
 Open tempfile for Input as #1
 Do While Eof(1)<>-1
 Input #1, acctno(x)
 msgtext=msgtext & Chr(10) & acctno(x)
 x=x+1
 Loop
 MsgBox msgtext
 Close #1
 Kill "C:\TEMP001"
End Sub
```

**See Also**     Close     Print  
              Open     Put



# WriteTestCaseResult

Utility Command

»»SQA»

This command is obsolete and should not be used. It continues to be supported to maintain the upward compatibility of your existing scripts.

## Year

Function

**Description** Returns the year component of a date or date/time value.

**Syntax** **Year**(*date*)

| Syntax Element | Description                |
|----------------|----------------------------|
| <i>date</i>    | A date or date/time value. |

**Comments** *Year* returns a year between 100 and 9999, inclusive.

*Year* accepts valid date and date/time formats, including numbers and strings, and will attempt to convert the input value to a date value. Examples of valid *date* values:

```
12/27/98
12/27/98 11:53:49 AM
Dec 27 1948
27 Dec 1948
December 27, 1998
```

The return value is a Variant of VarType 2 (Integer). If the value of *date* is NULL, a Variant of VarType 1 (Null) is returned.

With this function, a two-digit year is converted to a four-digit year, as follows:

- ▶ 00 through 29 is converted to 2000 through 2029
- ▶ 30 through 99 is converted to 1930 through 1999

When exchanging data information with external data sources or external programs, you should use double-precision floating point numbers or data strings with at least four characters for identifying the century.

Year

**Example**      **This example returns the year for the current date.**

```
Sub main
 Dim nowyear
 nowyear=Year(Now)
 MsgBox "The current year is: " &nowyear
End Sub
```

**See Also**

|                |               |
|----------------|---------------|
| Date function  | Month         |
| Date statement | Now           |
| Day            | Second        |
| Hour           | Time function |
| Minute         | Weekday       |

▶ ▶ ▶ Appendixes



## ▸ ▸ ▸ A p p e n d i x A

# SQABasic Syntax Summary

### Arguments

Arguments are separated by commas. Arguments are sometimes enclosed in parentheses, as follows:

**Function** arguments are always enclosed in parentheses. See the *Function* section for more information.

**Sub procedure** arguments are not enclosed in parentheses *unless* you use the Call statement. See the *Sub Procedure* section for more information.

If you're passing an argument *by value*, enclose that particular argument in parentheses:

```
Call MySub((x),y)
```

### Array Dimensions

When declaring an array, list the array dimensions after the array name. Array dimensions are separated by commas and enclosed in parentheses:

```
Dim arrayname (6, 8, 500) As Integer
```

### Array Elements

A particular element in an array is specified through the index value of each dimension:

```
Dim MyArray(10, 50) As String ' First declare the array
Dim x, y
...
' Now check every element for an empty string
For x = 0 to 10
 For y = 0 to 50
 If MyArray(x,y) = "" then
 GoTo ErrorRoutine
 End If
 Next y
Next x
```

## Array Subscripts

Typically, only the subscript that sets the upper bound of an array dimension is specified (as shown in *Array Dimensions* above). When the lower-bound subscript is omitted, the lower bound defaults to either 0 or 1 (depending on the value of the `Option Base` statement).

However, both the lower-bound and the upper-bound subscript can be specified, as follows:

```
Dim arrayname (3 To 6, -8 To 8, 1 To 500) as Integer
```

## Comments

Comments are prefixed by an apostrophe ( ' ) or the statement `REM`. The SQABasic compiler ignores comments:

```
Const SIZE = 10 ' Set the size constant
```

**NOTE:** The metacommands `'$CStrings`, `'$Include`, and `'$NoCStrings` are exceptions. Even though their names begin with an apostrophe, the SQABasic compiler considers them to be commands, not comments.

## Context Notation

Object context can be set through backslash ( \ ) and dot-backslash ( . \ ) notation. These characters appear in the recognition method (*recMethod*) argument of commands and are delimited by semicolons ( ; ):

```
' Requires MyGrid to be in a window named MyWindow
"\;Name=MyWindow;\;Name=MyGrid;\;Name=MyColumn"

' MyGrid can be in whatever window is the current context window
".\;Name=MyGrid;\;Name=MyColumn"
```

## Functions

Functions return a value. They have a slightly different calling syntax than sub procedures:

```
' Use this form to retrieve the return value
ReturnValue=MyFunction([argument1,argumentn])

' Use this form to ignore the return value
Call MyFunction(argument1,argumentn)
```

## Labels

Labels allow you to jump to a particular line of code.

A label has the same format as any other SQABasic name. Keywords (such as command names) are reserved words and are not valid labels.

To be recognized as a label, a name must begin in the first column of a line of code, and must be immediately followed by a colon (:).

Use the `GoTo` statement to jump to the label:

```
GoTo MyLabel
.
.
.
MyLabel:
```

## Line Continuation Syntax

Line continuation syntax allows long statements to extend to the next physical line. Line continuation syntax consists of a space character followed by an underscore character (`_`):

```
Dim trMonth As Integer, _ ' Month of transaction
 trYear As Integer ' Year of transaction
```

Note that you can add a comment after the underscore.

## Line Numbers

Line numbers are not supported in SQABasic.

## Names

An SQABasic name (such as variable and label names) must start with a letter (A through Z, a through z). The remaining part of a name can also contain numeric digits (0 through 9) or an underscore character (`_`). A name cannot be more than 40 characters in length. Type-declaration characters are not considered part of a name.

## Parameters (*parameters\$*) Argument

Parameter values are used in many user action and verification point commands. If more than one parameter is listed, separate them with semicolons (`;`):

```
Result = CheckBoxVP (CompareProperties, "Text=Read Only", _
 "CaseID=VPTEN;Wait=6,30")
```

## Recognition Method (*recMethod\$*) Argument

The *recMethod\$* arguments are used in many user action and verification point commands. If more than one recognition method is listed, separate them with semicolons (;):

```
Result = FileVP (Compare, _
 "File1= MYPROG.EXE;File2=C:\OLDPROG.EXE", _
 "CaseID=FCMYPROG")
```

## Strings

Strings are enclosed in double quotation marks ("):

```
CustName = "Robert Lentz"
```

## Sub Procedures

Sub procedures don't return a value. They have a slightly different calling syntax than functions:

```
' If using this form, you need parentheses around your arguments
Call MySubProc (argument1, argumentn)
```

```
' If using this form, omit the parentheses
MySubProc argument1,argumentn
```

## Variables of User-Defined Type

Variables of User-Defined type use dot notation to separate the name of the variable from its elements:

```
Cust.fName = "John" ' Customer's first name
Cust.lName = "Smith" ' Customer's last name
Cust.ID = 12345 ' Customer's unique ID
```



## ▶ ▶ ▶ Appendix B

# Trappable Error Codes

The following table lists the runtime errors SQABasic returns. These errors can be trapped through the `On Error` statement.

Use the `Err` function to query the error code, and use the `Error` function to query the error text.

| Code | Error Text                  | Code | Error Text                            |
|------|-----------------------------|------|---------------------------------------|
| 5    | Illegal function call       | 70   | Permission denied                     |
| 6    | Overflow                    | 71   | Disk not ready                        |
| 7    | Out of memory               | 74   | Can't rename with different drive     |
| 9    | Subscript out of range      | 75   | Path/File access error                |
| 10   | Duplicate definition        | 76   | Path not found                        |
| 11   | Division by zero            | 91   | Object variable set to Nothing        |
| 13   | Type Mismatch               | 93   | Invalid pattern                       |
| 14   | Out of string space         | 94   | Illegal use of NULL                   |
| 19   | No Resume                   | 102  | Command failed                        |
| 20   | Resume without error        | 429  | Object creation failed                |
| 28   | Out of stack space          | 438  | No such property or method            |
| 35   | Sub or Function not defined | 439  | Argument type mismatch                |
| 48   | Error in loading DLL        | 440  | Object error                          |
| 52   | Bad file name or number     | 901  | Input buffer would be larger than 64K |
| 53   | File not found              | 902  | Operating system error                |

## Trappable Error Codes

| Code | Error Text             | Code | Error Text                            |
|------|------------------------|------|---------------------------------------|
| 54   | Bad file mode          | 903  | External procedure not found          |
| 55   | File already open      | 904  | Global variable type mismatch         |
| 58   | File already exists    | 905  | User-defined type mismatch            |
| 61   | Disk full              | 906  | External procedure interface mismatch |
| 62   | Input past end of file | 907  | Push button required                  |
| 63   | Bad record number      | 908  | Module has no MAIN                    |
| 64   | Bad file name          | 910  | Dialog box not declared               |
| 68   | Device unavailable     |      |                                       |

▶ ▶ ▶ A p p e n d i x C

## Object Scripting Status Codes

The following table contains the Integer values that are returned from the SQABasic Object Scripting commands:

| Numeric | Literal                              | Description                                                                                                                                                                                                                                       |
|---------|--------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0       | <code>sqaSuccess</code>              | Command executed successfully.                                                                                                                                                                                                                    |
| 1001    | <code>sqaNoObjectSpecified</code>    | <i>Recognition Method</i> is empty.                                                                                                                                                                                                               |
| 1002    | <code>sqaInvalidRecString</code>     | Invalid syntax in <i>Recognition Method</i> .                                                                                                                                                                                                     |
| 1003    | <code>sqaObjectNotFound</code>       | The specified object couldn't be found.                                                                                                                                                                                                           |
| 1004    | <code>sqaNoPropertySpecified</code>  | The <i>property\$</i> argument is empty.                                                                                                                                                                                                          |
| 1005    | <code>sqaPropertyNotFound</code>     | The specified property couldn't be found.                                                                                                                                                                                                         |
| 1006    | <code>sqaErrorGettingProperty</code> | An error occurred while getting a property value.                                                                                                                                                                                                 |
| 1007    | <code>sqaArraysNotSupported</code>   | No index was specified for an array of property values.                                                                                                                                                                                           |
| 1008    | <code>sqaPropertyIsNotArray</code>   | An index was specified for a property that does not contain an array of values.                                                                                                                                                                   |
| 1009    | <code>sqaPropertyHasNoValue</code>   | The property has no value or is currently not applicable.<br>A property that has no value is not the same as a property with a value of 0 or null. For example, the <code>ItemSelected</code> property for an empty list box has no value.        |
| 1010    | <code>sqaPropertyNotSupported</code> | The property is not supported.<br>This status typically results when the operating system you're testing on doesn't support a property — for example, the <code>NumbersOnly</code> property for an edit box isn't supported under 16-bit Windows. |

▶ ▶ ▶

## Object Scripting Status Codes

▶ ▶ ▶

| Numeric | Literal                               | Description                                                                                                                                                                                                                          |
|---------|---------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1011    | <code>sqaUnableToConvertString</code> | Can't convert the specified property value to a <code>String</code> .                                                                                                                                                                |
| 1012    | <code>sqaPropertyIsReadOnly</code>    | The property value is read-only and can't be modified.                                                                                                                                                                               |
| 1013    | <code>sqaInvalidDataType</code>       | The data type of the property can't be converted to a <code>Variant</code> . With <code>SQASetProperty</code> commands, the type ( <code>VarType</code> ) of <code>Variant</code> can't be converted to a property.                  |
| 1014    | <code>sqaInvalidPropertyValue</code>  | Invalid property value.<br>For example, this error occurs if you try to set the <code>State</code> property for a check box to a value that is not one of the property's choices ( <code>Checked</code> or <code>Unchecked</code> ). |
| 1015    | <code>sqaIndexOutOfBounds</code>      | The array index is not within the bounds of the array.                                                                                                                                                                               |
| 1016    | <code>sqaTimeout</code>               | The specified object couldn't be found within the specified time, or (with <code>SQAWaitForPropertyValue</code> ) was not equal to the specified time.                                                                               |
| 1017    | <code>sqaNoMethodSpecified</code>     | No method was specified in <i>Object Method</i> .                                                                                                                                                                                    |
| 1018    | <code>sqaMethodNotFound</code>        | The method specified in <i>Object Method</i> doesn't exist.                                                                                                                                                                          |
| 1019    | <code>sqaErrorInvokingMethod</code>   | An error occurred while attempting to execute the method. A likely cause is either that the method doesn't exist or the method arguments are invalid.                                                                                |
| 1020    | <code>sqaErrorSettingProperty</code>  | An error occurred while setting a property value with <code>SQASetProperty</code> .                                                                                                                                                  |
| 1030    | <code>sqaOutOfMemory</code>           | There isn't enough memory to run this command.                                                                                                                                                                                       |
| 1031    | <code>sqaUserAbort</code>             | The operation was canceled — for example, by pressing F11.                                                                                                                                                                           |
| 1040    | <code>sqaUnknownError</code>          | An unknown error occurred.                                                                                                                                                                                                           |

## ► ► ► Appendix D

# Derived Trigonometric Functions

Many trigonometric operations can be constructed from built-in functions:

| <b>Function</b>         | <b>Computed By</b>                                                                 |
|-------------------------|------------------------------------------------------------------------------------|
| Secant                  | $\text{Sec}(x) = 1/\text{Cos}(x)$                                                  |
| CoSecant                | $\text{CoSec}(x) = 1/\text{Sin}(x)$                                                |
| CoTangent               | $\text{CoTan}(x) = 1/\text{Tan}(x)$                                                |
| ArcSine                 | $\text{ArcSin}(x) = \text{Atn}(x/\text{Sqr}(-x*x+1))$                              |
| ArcCosine               | $\text{ArcCos}(x) = \text{Atn}(-x/\text{Sqr}(-x*x+1))+1.5708$                      |
| ArcSecant               | $\text{ArcSec}(x) = \text{Atn}(x/\text{Sqr}(x*x-1))+\text{Sgn}(x-1)*1.5708$        |
| ArcCoSecant             | $\text{ArcCoSec}(x) = \text{Atn}(x/\text{Sqr}(x*x-1))+(\text{Sgn}(x)-1)*1.5708$    |
| ArcCoTangent            | $\text{ArcTan}(x) = \text{Atn}(x)+1.5708$                                          |
| Hyperbolic Sine         | $\text{HSin}(x) = (\text{Exp}(x)-\text{Exp}(-x))/2$                                |
| Hyperbolic Cosine       | $\text{HCos}(x) = (\text{Exp}(x)+\text{Exp}(-x))/2$                                |
| Hyperbolic Tangent      | $\text{HTan}(x) = (\text{Exp}(x)-\text{Exp}(-x))/(\text{Exp}(x)+\text{Exp}(-x))$   |
| Hyperbolic Secant       | $\text{HSec}(x) = 2/(\text{Exp}(x)+\text{Exp}(-x))$                                |
| Hyperbolic CoSecant     | $\text{HCoSec}(x) = 2/(\text{Exp}(x)-\text{Exp}(-x))$                              |
| Hyperbolic Cotangent    | $\text{HCotan}(x) = (\text{Exp}(x)+\text{Exp}(-x))/(\text{Exp}(x)-\text{Exp}(-x))$ |
| Hyperbolic ArcSine      | $\text{HArcSin}(x) = \text{Log}(x+\text{Sqr}(x*x+1))$                              |
| Hyperbolic ArcCosine    | $\text{HArcCos}(x) = \text{Log}(x+\text{Sqr}(x*x-1))$                              |
| Hyperbolic ArcTangent   | $\text{HArcTan}(x) = \text{Log}((1+x)/(1-x))/2$                                    |
| Hyperbolic ArcSecant    | $\text{HArcSec}(x) = \text{Log}((\text{Sqr}(-x*x+1)+1)/x)$                         |
| Hyperbolic ArcCoSecant  | $\text{HArcCoSec}(x) = \text{Log}((\text{Sgn}(x)*\text{Sqr}(x*x+1)+1)/x)$          |
| Hyperbolic ArcCoTangent | $\text{HArcCoTan}(x) = \text{Log}((x+1)/(x-1))/2$                                  |

## Derived Trigonometric Functions

## Mouse Actions

Mouse *click* actions occur when you click or double-click any of the mouse buttons while recording. The action written to the script depends on which button was clicked and what combination of SHIFT, CTRL, and ALT keys was held down at the time of the click. The same is true for mouse *drag* actions.

For example, if you double-click the left mouse button on an object while holding the SHIFT key down, a `Shift_DblClick` action is recorded. If you press the right mouse button down and drag an object while holding the CTRL and ALT keys down, a `CtrlAlt_Right_Drag` action is recorded.

Mouse actions using the left mouse button do not contain the word `Left` because the left button is the default for mouse actions. The button is specified only if the `Middle` or `Right` button is used in the action.

When a mouse click occurs on a check box, label, push button, or radio button, SQA Robot does not record coordinates of the click because the actual position is not important. When playing back a recorded click on one of these objects, SQA Robot clicks in the center of the specified object.

When a mouse click occurs on a combo list box or list box, SQA Robot records the selection information, not the coordinates of the click. For example, a mouse click on a list box would indicate the selected item, not the coordinates where the click occurred. When playing back a recorded click on one of these objects, SQA Robot selects the specified item, regardless of its position in the list or the dimensions of the box.

## Mouse Actions

### MouseClicked Actions

These are the valid values for a *MouseClicked* action:

|                           |                              |
|---------------------------|------------------------------|
| Click                     | DoubleClick                  |
| Middle_Click              | Middle_DblClick              |
| Right_Click               | Right_DblClick               |
| Shift_Click               | Shift_DblClick               |
| Shift_Middle_Click        | Shift_Middle_DblClick        |
| Shift_Right_Click         | Shift_Right_DblClick         |
| Ctrl_Click                | Ctrl_DblClick                |
| Ctrl_Middle_Click         | Ctrl_Middle_DblClick         |
| Ctrl_Right_Click          | Ctrl_Right_DblClick          |
| Alt_Click                 | Alt_DblClick                 |
| Alt_Middle_Click          | Alt_Middle_DblClick          |
| Alt_Right_Click           | Alt_Right_DblClick           |
| ShiftCtrl_Click           | ShiftCtrl_DblClick           |
| ShiftCtrl_Middle_Click    | ShiftCtrl_Middle_DblClick    |
| ShiftCtrl_Right_Click     | ShiftCtrl_Right_DblClick     |
| ShiftAlt_Click            | ShiftAlt_DblClick            |
| ShiftAlt_Middle_Click     | ShiftAlt_Middle_DblClick     |
| ShiftAlt_Right_Click      | ShiftAlt_Right_DblClick      |
| CtrlAlt_Click             | CtrlAlt_DblClick             |
| CtrlAlt_Middle_Click      | CtrlAlt_Middle_DblClick      |
| CtrlAlt_Right_Click       | CtrlAlt_Right_DblClick       |
| ShiftCtrlAlt_Click        | ShiftCtrlAlt_DblClick        |
| ShiftCtrlAlt_Middle_Click | ShiftCtrlAlt_Middle_DblClick |
| ShiftCtrlAlt_Right_Click  | ShiftCtrlAlt_Right_DblClick  |

### MouseDown Actions

These are the valid values for a *MouseDown* action:

|                   |                          |
|-------------------|--------------------------|
| Left_Drag         | ShiftCtrl_Drag           |
| Right_Drag        | ShiftCtrl_Middle_Drag    |
| Middle_Drag       | ShiftCtrl_Right_Drag     |
| Shift_Drag        | ShiftAlt_Drag            |
| Shift_Middle_Drag | ShiftAlt_Middle_Drag     |
| Shift_Right_Drag  | ShiftAlt_Right_Drag      |
| Ctrl_Drag         | CtrlAlt_Drag             |
| Ctrl_Middle_Drag  | CtrlAlt_Middle_Drag      |
| Ctrl_Right_Drag   | CtrlAlt_Right_Drag       |
| Alt_Drag          | ShiftCtrlAlt_Drag        |
| Alt_Middle_Drag   | ShiftCtrlAlt_Middle_Drag |
| Alt_Right_Drag    | ShiftCtrlAlt_Right_Drag  |



## ▶ ▶ ▶ Glossary

action object – In TestFactory, an object in the application map that represents an action to which a control in the application responds. Typical actions are mouse left-click, mouse right-click, and mouse left-double-click; the corresponding action objects in the application map are LeftClick, RightClick, and LeftDoubleClick.

ActiveX control – A reusable software control that takes advantage of Object Linking and Embedding (OLE) and Component Object Modeling (COM) technologies. Developers can use ActiveX controls to add specialized functions to applications, software development tools, and Web pages. Robot can test ActiveX controls in applications.

actual results – In a functional test, the outcome of testing an object through a verification point in a GUI script. Actual results that vary from the recorded baseline results are defects or intentional changes in the application. See also *baseline results*.

Administrator – See *Rational Administrator*.

Agent computer – In LoadTest, a computer that has the Rational Agent software installed and that plays back a virtual user or GUI script. In a LoadTest schedule, you can identify the Agent computer on which to run a script. See also *Rational Agent*.

API recording – In Robot, a virtual user recording method that captures API calls between a specific client application and a server. These calls are captured on the client computer.

application map – In TestFactory, a hierarchical list of controls and actions in the application-under-test, as well as the states of the application-under-test and the transitions between those states. An application map can include UI objects and action objects, as well as TestFactory objects such as Pilots, Test Suites, and scripts.

application-under-test – The software being tested. See also *system-under-test*.

Asset Browser – A window that displays testing resources such as builds, queries, scripts, schedules, reports, report output, and logs. The Asset Browser is available in TestManager and LoadTest.

AUT – See *application-under-test*.

## Glossary

automated testing – A testing technique in which you use software tools to replace repetitive and error-prone manual work. Automated testing saves time and enables a reliable, predictable, and accurate testing process.

AutoPilot – In TestFactory, a tool for running scripts, Test Suites, and Pilots. The scripts and Test Suites can run on your local computer or on computers in the Test Lab. The Pilots run on your local computer, and the scripts they generate can run on your local computer or on computers in the Test Lab.

base state – In TestFactory, the known, stable state in which you expect the application-under-test to be at the start of each script segment. See also *script segment*.

baseline results – In a functional test, the outcome of testing an object through a verification point in a GUI script. The baseline results become the expected state of the object during playback of the script. Actual test results that vary from the baseline results are defects or intentional changes in the application. See also *actual results*.

best script – In TestFactory, an optimized script generated by a Pilot. A best script contains the fewest number of script segments that provide the most coverage of the source code or user interface in the application-under-test.

breakpoint – A feature of the Robot debugger. When you assign a breakpoint to a line of code, and then run the script in the debugger environment, the script stops executing at that line of code. Control returns to you, and the breakpoint line is displayed. From here you can view variables, perform other debugging activities, and continue executing the script.

build – A version of the application-under-test. Typically, developers add new features or enhancements to each incremental build. As team members test a build, they enter defects against those features that do not behave as expected. You use TestManager to define and manage builds.

built-in data test – A data test that comes with Robot and is used with the Object Data verification point. A data test uses a specific property of the object, in conjunction with other parameters, to determine the data to capture. Although built-in data tests cannot be edited, renamed, or deleted, they can be copied and then edited, and they can be viewed. See also *custom data test*.

ClearQuest – See *Rational ClearQuest*.

client/server – An architecture for cooperative processing in which the software tasks are split between server tasks and client tasks. The client computer sends requests to the server, and the server responds.

code coverage – In TestFactory, the percentage of code that is tested by a script. This percentage is based on the portion of the code that a script touches, relative to all code in the application-under-test. A Pilot can use code coverage to determine the best script for a run. See also *UI coverage*.

command ID – In LoadTest's VU language, an identifier for a command. Robot automatically assigns a unique command ID, composed of an alphanumeric prefix and a three-digit number, to each emulation command. Because command IDs appear in both the virtual user script and the LoadTest report output, they enable you to determine the relationship between an emulation command and its response times.

command ID prefix – In LoadTest, a prefix for a unique emulation command ID. The prefix defaults to the script name (up to the first seven characters). However, you can define the prefix in the Generator tab of the Virtual User Record Options dialog box.

custom data test – A customer-defined data test used with the Object Data verification point. A data test uses a specific property of the object, in conjunction with other parameters, to determine the data to capture. Custom data tests are created within your organization and are stored in the repositories that were active when they were created. They can be edited, renamed, and deleted. See also *built-in data test*.

data test – A test that captures the data of an object with the Object Data verification point. See also *built-in data test* and *custom data test*.

datapool – A source of test data that GUI scripts and virtual user scripts can draw from during playback. You can automatically generate datapools using TestManager, or you can import datapool data from other sources such as your database.

dependency – In LoadTest, a method of coordinating an object in a schedule with an event. For example, if the script Query is dependent upon the script Connect, then Connect must finish executing before Query can begin executing. See also *event*.

distributed architecture – Architecture in which computer systems work together and communicate with each other across LAN, WAN, or other types of networks. A client/server system is an example of distributed architecture.

distributed functional test – In LoadTest, a test that uses multiple Agent computers to execute multiple GUI scripts written in the SQABasic language.

dynamic load balancing selector – A type of selector in a LoadTest schedule. Items in the selector, such as scripts, are executed according to a weight you set.

## Glossary

emulation commands – VU language statements or commands that emulate client activity, evaluate the server's responses, and perform communication and timing operations. LoadTest stores the results of emulation commands in a log file, which you can view from the LogViewer.

emulation functions – VU language functions that emulate client activity and evaluate the server's responses. Unlike emulation commands, emulation functions do not perform communication and timing operations, and they are not logged.

environment control commands – VU language commands that let you control a virtual user's environment by changing the VU environment variables. For example, you can set the level of detail that is logged or the number of times that virtual users attempt to connect to a server.

event – An item in a LoadTest schedule upon which another item is dependent. For example, if the script Connect sets an event and the script Query depends on this event, Connect must finish executing before Query can begin executing. See also *dependency*.

external script – A script that runs a program created with any tool. You plan and run external scripts in TestManager.

fixed user group – In LoadTest, a group that contains a scalable number of users. When you create a fixed user group, you indicate the maximum number of users that you will run in the group. Typically, you use fixed user groups in functional tests, which do not add a workload to the system.

flow control statements – In the VU and SQABasic languages, statements that let you add conditional execution structures and looping structures to a script.

functional test – A test to determine whether a system functions as intended. Functional tests are performed on GUI objects and objects such as hidden DataWindows and Visual Basic hidden controls.

Grid Comparator – The Robot component for reviewing, analyzing, and editing data files for text and numeric verification points in grid formats. The Grid Comparator displays the differences between the recorded baseline data and the actual data captured during playback.

GUI script – A type of script written in the SQABasic language. It contains GUI actions such as keystrokes and mouse clicks. Typically, a GUI script also contains verification points for testing objects over successive builds of the application-under-test.

GUI user – The type of user that is emulated when a GUI script is executed. Only one GUI user at a time can run on a computer.

hidden object – An object that is not visible through the user interface. Hidden objects include objects with a visible property of False and objects with no GUI component.

IDE – Integrated Development Environment. This environment consists of a set of integrated tools that are used to develop a software application. Examples of IDEs supported by Robot include Oracle Forms, PowerBuilder, Visual Basic, and Java.

Image Comparator – The Robot component for reviewing and analyzing bitmap image files for Region Image and Window Image verification points. The Image Comparator displays differences between the recorded baseline image and the actual image captured during playback. The Image Comparator also displays unexpected active windows that appear during playback.

instrumentation – In TestFactory, the process of inserting code coverage counters into the application-under-test. These counters record how much code is executed during a script run. See also *object code instrumentation* and *source code instrumentation*.

load – See *workload*.

load balancing – See *workload balancing*.

LoadTest – See *Rational LoadTest*.

log – A repository object that contains the record of events that occur while playing back a script or running a schedule. A log includes the results of all verification points executed as well as performance data that can be used to analyze the system's performance.

LogViewer – See *Rational LogViewer*.

low-level recording – A recording mode that uses detailed mouse movements and keyboard actions to track screen coordinates and exact timing. During playback, all actions occur in real time, exactly as recorded.

manual script – A set of testing instructions to be run by a human tester. The script can consist of steps and verification points. You create manual scripts in TestManager.

Master computer – A computer that executes LoadTest. From this computer, you create, run, and monitor schedules. When the run is finished, you use it to analyze test results.

mix-ins – See *Pilot mix-ins*.

network recording – In Robot, a virtual user recording method that records packet-level traffic. This traffic is captured on the wire.

next available selector – In LoadTest schedules, a selector that distributes each item such as a script, delay, or other selector to an available computer or virtual user. This type of selector is used in a GUI schedule. The next available selector parcels out the items sequentially, based on which computers or virtual users are available.

object – An item on a screen, such as a window, dialog box, check box, label, or command button. An object has information (properties) associated with it and actions that can be performed on it. For example, information associated with the window object includes its type and size, and actions include clicking and scrolling. In some development environments, a term other than *object* is used. For example, the Java environment uses *component*, and the HTML environment uses *element*.

object code instrumentation – In TestFactory, the process of inserting code coverage counters into the executable file of the application-under-test. These counters record how much of the program a script tests. See also *instrumentation* and *source code instrumentation*.

Object-Oriented Recording® – A script recording mode that examines objects in the application-under-test at the Windows layer. Robot uses internal object names to identify objects, instead of using mouse movements or absolute screen coordinates.

Object Properties Comparator – The Robot component that you use to review, analyze, and edit the properties of objects captured by an Object Properties verification point. The Object Properties Comparator displays differences between recorded baseline data and the actual data captured during playback.

Object Scripting commands – A set of SQABasic commands for accessing an application's objects and object properties. You add Object Scripting commands manually when editing a script.

Object Testing® – A technology used by Robot to test any object in the application-under-test, including the object's properties and data. Object Testing lets you test standard Windows objects and IDE-specific objects, whether they are visible in the interface or hidden.

OCI – Object Code Insertion. The Rational technology used in TestFactory to instrument object code and measure how much of the application-under-test a script tests. See also *code coverage* and *object code instrumentation*.

performance test – A test that determines whether a multi-client system performs within user-defined standards under varying loads. Performance tests are always run from a schedule in LoadTest.

Pilot – In TestFactory, a tool for generating scripts automatically.

**Pilot mix-ins** – In TestFactory, a list of Pilots that are executed on a random basis during the run of a lead Pilot. Mix-ins are useful for randomly testing multiple areas of the application-under-test. To make tests more realistic, you can combine mix-ins and scenarios.

**Pilot scenario** – An ordered list of Pilots that are executed during the run of a Pilot. A Pilot scenario is useful for testing UI objects that need to be exercised in a specific order. To make tests more realistic, you can combine scenarios and mix-ins.

**project** – A collection of data, including test assets, defects, requirements, and models, that can facilitate the development and testing of one or more software components.

**proxy recording** – In Robot, a virtual user recording method that captures the client/ server conversation on the network wire rather than on the client computer. Proxy recording allows Robot to capture network packets that are not visible to it during network recording — for example, if the client and server are in different network segments.

**query** – A request for information stored in the repository. A query consists of a filter and several visible attributes — the columns of data to display, the width of the column, and the sort order.

**random selector** – A type of selector in a LoadTest schedule. Items in the selector, such as scripts, are randomly executed. Random selectors can be with replacement, where the odds are the same, or without replacement, where the odds change with each iteration.

**Rational Administrator** – The component for creating and maintaining repositories, projects, users, groups, computers, and SQL Anywhere servers.

**Rational Agent** – The LoadTest software that resides on a shared network drive and runs on each computer where testing occurs. The entries specified in a schedule play back on the Agent computer, which reports on their progress and status as they run. See also *Agent computer*.

**Rational ClearQuest** – The Rational product for tracking and managing defects and change requests throughout the development process. With ClearQuest, you can manage every type of change activity associated with software development, including enhancement requests, defect reports, and documentation modifications.

**Rational LoadTest** – The Rational Test component for running performance, stress, scalability, multi-user, and distributed functional tests on multiple Agents connected by a network. With LoadTest, you can initiate test runs and monitor tests from a master computer that manages the test process. LoadTest is available only in Rational Suite PerformanceStudio.

## Glossary

Rational LogViewer – The Robot component for displaying logs, which contain the record of events that occur while playing back a script or running a schedule. Also, the component from which you start the four Comparators.

Rational PerformanceArchitect – The Rational component that lets you test the performance of COM/DCOM applications. With Rational PerformanceArchitect, you can create a Rose sequence or collaboration diagram, convert it to a virtual user script, and then use Rational Suite PerformanceStudio to edit the script and run the performance tests.

Rational repository – A database that stores application testing information, such as test requirements, scripts, and logs. All Rational Suite TestStudio and Rational Suite PerformanceStudio products and components on your computer update and retrieve data from the same connected repository. A repository can contain either a Microsoft Access or a Sybase SQL Anywhere database.

Rational RequisitePro – The Rational product for organizing, managing, and tracking the changing requirements of your system.

Rational Robot – The Rational product for recording, playing back, debugging, and editing scripts.

Rational SiteCheck – The Robot component for managing your intranet or World Wide Web site. You can use SiteCheck to visualize the structure of your Web site, and you can use it with Robot to automate Web site testing.

Rational Synchronizer – The Rational tool that ensures the consistency of data across several Rational products.

Rational TestAccelerator – An agent application that executes scripts. TestFactory uses computers running TestAccelerator as remote machines on which to run automated distributed tests.

Rational TestFactory – The Rational Test component for mapping an application-under-test and generating scripts automatically. TestFactory is available in Rational Suite TestStudio and Rational Suite PerformanceStudio.

Rational TestManager – The Robot component for managing the overall testing effort. You use it to define and store information about test documents, requirements, scripts, schedules, and sessions.

Report Layout Editor – The TestManager component for customizing the layout of reports.

repository – See *Rational repository*.

RequisitePro – See *Rational RequisitePro*.

Robot – See *Rational Robot*.



scalable user group – In LoadTest, a group that contains a varying number of users. When you create a scalable user group, you assign it a percentage of the total workload. Assume you have a scalable user group that is 50 percent of the workload. If you run a test with 10 users, the group will contain 5 users. If you run a test with 100 users, the group will contain 50 users.

scenario – In LoadTest, a modular group of scripts and other items in a schedule that is used by more than one user group. A scenario can contain scripts, delays, and synchronization points.

scenario – See *Pilot scenario*.

schedule – In LoadTest, structure that you create to specify how scripts should be played back. A schedule can contain GUI scripts and virtual user scripts, and can indicate the number of times to repeat a script and the computer on which the script will run. In performance testing, a schedule is used to create a workload. In distributed functional testing, a schedule is used to distribute scripts among various computers.

script – A set of instructions used to navigate through and test an application. You can generate scripts in a variety of ways. You can use Robot to record scripts used in functional testing and performance testing. You can also use TestManager to create and manage manual scripts, and to manage external scripts created with a third-party testing tool. A script can have properties associated with it, such as the purpose of the script and requirements for the script. See also *external script*, *GUI script*, *manual script*, and *virtual user script*.

script outline – In TestFactory, the readable version of a script. A script outline contains a description of the actions that Robot performs while running the script.

script segment – In TestFactory, a section of a script that tests a particular element of product functionality. A Pilot generates a script segment by starting the application-under-test in a base state, navigating through the part of the product that you are testing, and returning the application-under-test to the base state. See also *base state*.

seed – An initial number fed to a random number generator. Using the same seed produces the same series of random numbers. In LoadTest, you use seeds to generate think times.

selector – An item that you insert in a LoadTest schedule to indicate how often and in what order to run scripts.

sequential selector – In a LoadTest schedule, a type of selector that executes each script, delay, or other item in the same order in which it appears in the schedule.

## Glossary

session – In virtual user recording, one or more scripts that you record from the time you begin recording until the time you stop recording. Typically, the scripts in a session represent a logical flow of tasks for a particular user, with each script representing one task. For example, a session could be made up of three scripts: *login*, *testing*, and *logout*. In TestFactory, a session is the period of time that the TestFactory application or a window is open.

shared variable – An integer variable that multiple scripts and multiple virtual users can read and write to. You can see the value of a shared variable while monitoring a LoadTest schedule. For example, you can set a shared variable as a flag to end a playback session. Each script can check the flag to see if the session should end. When that flag is set, exit tasks can be performed.

shell script – A script that calls or groups several other GUI scripts and plays them back in sequence. Shell scripts provide the ability to create comprehensive tests and then store the results in a single log.

SiteCheck – See *Rational SiteCheck*.

source code instrumentation – In TestFactory, the process of inserting code into the source code of the application-under-test. This code measures how much of the source code a script tests. See also *instrumentation* and *object code instrumentation*.

SQABasic – The Robot scripting language for recording GUI actions and verifying GUI objects. SQABasic contains most of the syntax rules and core commands that are contained in the Microsoft Basic language. In addition, SQABasic has commands that are specifically designed for automated testing. See also *VU*.

stable load – In LoadTest, a condition that occurs when a specified number of virtual users have logged on to the system-under-test and are active. When the stable load criterion is met, LoadTest begins measuring the load.

streak – When running a virtual user schedule in LoadTest, a series of successes or failures for emulation commands. You can see a streak while monitoring a schedule.

structural test – A test to determine whether the structure of a Web site is consistent and complete. A structural test ensures that an application's interdependent objects are properly linked together. You perform a structural test using SiteCheck.

synchronization point – In LoadTest, a place where emulated virtual users stop and wait until all other synchronized users reach that point. When all users reach the synchronization point, they are released and continue executing.

Synchronizer – See *Rational Synchronizer*.

system tuning – In LoadTest, the process of optimizing a system’s performance by changing hardware resources and software configuration parameters while using a constant workload.

system-under-test – The system being tested. This includes the computers and any software that can generate a load on the system, networks, user interfaces, CPUs, and memory. See also *application-under-test*.

test assets – The resources that facilitate the planning or development phases of the testing effort. Examples of test assets include scripts, schedules, sessions, test documents, and test requirements.

test development – The process of developing tests to verify the operation of a software application. This includes creating scripts that verify that the application-under-test functions properly. Test development lets you establish the baseline of expected behavior for the application-under-test.

test documents – Test plans, project schedules, resource requirements, and any other documents that are important to your project. You develop your test documents using your own word processing or scheduling program; you then reference the name and location of the document in TestManager. This lets members of the test and development team locate documents quickly.

Test Lab – A collection of computers on which TestAccelerator is running. In TestFactory, you can distribute the scripts associated with a Pilot, a Test Suite, or the AutoPilot to run on computers in the Test Lab. See also *Rational TestAccelerator*.

Test Suite – In TestFactory, a tool for running a collection of scripts as a group.

TestAccelerator – See *Rational TestAccelerator*.

TestFactory – See *Rational TestFactory*.

TestManager – See *Rational TestManager*.

Text Comparator – The Robot component for reviewing, analyzing, and editing data files for text and numeric verification points in any format except grids. The Text Comparator displays the differences between the recorded baseline results and the actual results.

think time – In virtual user and GUI scripts, think times are delays that simulate a user’s pauses to type or think while using an application. With virtual user scripts, LoadTest calculates the think time at runtime, based on think time VU environment variables that are set in the script. You can set a maximum think time in Robot. With GUI scripts, Robot uses the actual delays captured between keystrokes, menu choices, and other actions.

transaction – In LoadTest, a logical unit of work performed against a server. For example, submitting a search query or submitting a completed form to a Web server are both transactions.

## Glossary

transaction rate – In LoadTest, the playback speed calculated as a function of number of transactions per unit of time. For example, if a script contains one transaction, and each script is started at half-second intervals, your transaction rate would be 2 per second.

transactor – In LoadTest, an item that you insert in a LoadTest schedule to indicate the number of user-defined transactions that a virtual user performs in a given time period.

UI coverage – In TestFactory, the percentage of objects in the application map that are tested by a Pilot-generated script. This percentage is the proportion of UI objects that the script touches, relative to all UI objects available to the Pilot. A Pilot can use UI coverage to determine the best script for a run. See also *code coverage*.

UI object properties – Attributes of object classes and UI objects that TestFactory uses to map applications and generate scripts.

unexpected active window – A window that appears during script playback that interrupts the script playback process and prevents the expected window from being active. For example, an error message generated by the application-under-test is an unexpected active window. You can view unexpected active windows in the Image Comparator.

user group – In LoadTest, a collection of users that execute similar tasks and generate the same basic workload. Accountants and data entry operators are examples of user groups.

verification – The process of comparing the test results from the current build of the software to its baseline results.

verification point – A point in an SQABasic script that confirms the state of one or more objects. During recording, a verification point captures object information from the application-under-test and stores it as the baseline. During playback, a verification point recaptures the object information and compares it to the baseline. In a manual script, a verification point is a question about the state of the application-under-test.

virtual user – In LoadTest, a type of user that is emulated when a virtual user script is executed. A computer can run multiple virtual users simultaneously.

virtual user script – A type of script written in the VU language. Virtual user scripts contain client/server requests and responses as well as user think times.

VU – The Robot scripting language for recording a client's requests to a server. VU provides most of the syntax rules and core commands available in the C programming language. In addition, VU has emulation commands and functions that are specifically designed for automated performance testing. See also *SQABasic*.

wait state – A delay or timing condition that handles time-dependent activities.

workload – In LoadTest, the set of all activities that users perform in an actual production setting of the system-under-test. You can use LoadTest to emulate a workload.

workload balancing – In LoadTest, the act of distributing activities so no one system or device becomes a bottleneck.

workload model – In LoadTest, the workload model is represented as a schedule. You can play back this schedule and analyze the response times.

## Glossary

## ▶ ▶ ▶ Index

### Symbols

- numeric operator, 3-12
- '\$CStrings metacommand, 6-64
- '\$Include metacommand, 6-218
- '\$NoCStrings metacommand, 6-328
- & string concatenation operator, 3-12
- \* numeric operator, 3-12
- \* wildcard, 4-17, 4-19
- .\ separator, 4-18
- .csv files, 5-12
- .dll files, 3-4, 4-28
- .lnk files, 6-501
- .rec files, 1-3, 4-2
  - as library files, 4-26
- .sbh files, 1-3, 4-29
- .sbl files, 1-3, 4-26
- .sbx files, 1-3, 4-2, 4-27
- .tpl files, 4-33
- / numeric operator, 3-12
- ; separator, 6-500, A-3, A-4
- ? wildcard, 4-17, 4-19
- \ escape character, 4-17, 4-20
- \ numeric operator, 3-12
- \ separator, 4-18
- ^ numeric operator, 3-12
- \_ line continuation character, A-3
- | separator, 6-364
- + numeric operator, 3-12
- + string concatenation operator, 3-12
- < comparison operator, 3-13
- = comparison operator, 3-13
- > comparison operator, 3-13
- > separator. *See* pointer separator

### A

- Abs function, 6-2
- absolute value, 6-2
- access Clipboard, 6-37
- action% argument, 4-8
- activate window, 6-6
- active window
  - AppActivate statement, 6-6
  - assigning context to, 4-16
  - Window user action command, 6-586
- ActiveX Test Control, 6-523
- actual data, 5-12, 5-15, 5-16
- actual data files, 4-7
  - ownership, 5-20
  - retrieving the location, 6-503
- additional property capture with Object Scripting, 5-7
- additions to Basic commands, 1-1, 1-3, 6-1
- alias, 6-89
- Analyzer utility, 5-21
- And logical operator, 3-13
- angle
  - cosine, 6-60
  - sine, 6-450
  - tangent, 6-547
- AnimateControl user action command, 6-2
- AnimateControlVP verification point command, 6-4
- ANSI characters, 6-35
- AppActivate statement, 6-6
- applications, starting
  - Shell command, 6-449
  - SQAShellExecute command, 6-501
  - StartApplication command, 6-522
  - StartJavaApplication command, 6-524
- arc tangent, 6-8
- arguments, 3-3, A-1
  - by-value and by-reference, 3-3
  - checking for presence of, 6-239
  - named, 3-4
  - passing, 6-22
  - user action and verification point commands, 4-8

## Index

- arrays, 3-10
  - command summary, 2-1
  - default lower-bound, 6-340
  - dimensions, 3-10, 6-98, 6-180, A-1
  - dynamic, 3-11
  - erasing, 6-139
  - global, 6-180
  - lower bound, 6-340
  - lower-bound subscripts, 6-289
  - of property values, 5-8
  - redimension, 6-417
  - retrieving property values as, 6-480, 6-482
  - size of, for property values, 6-483
  - subscripts, 3-10, 6-98, 6-180
  - upper bound, 6-573
  - upper-bound subscripts, 6-573

- Asc function, 6-7
- AscB, 6-7
- Assert statement, 6-7
- assign variables, 6-292
- assignment character, 6-494
- Atn function, 6-8
- attributes of files and directories, 6-175
- automatic script generation, 1-1

## B

- backslash ( \ ) and context, 4-18
- baseline, 1-3, 4-7
  - custom verification points, 5-12
- baseline data files, 4-7, 5-16
  - retrieving the location, 6-504
- BasicLib, 6-89
- Beep statement, 6-8
- Begin Dialog...End Dialog statement, 6-9
- Boolean data type, 3-7
- branching
  - GoTo statement, 6-182
  - On...GoTo statement, 6-336
- broken links, testing for, 6-579
- Browser utility command, 6-13

- browsers
  - default, for playback, 6-496
  - starting, 6-523
- Button statement, 6-16
- ButtonGroup statement, 6-17
- by-reference arguments, 3-3
- by-value arguments, 3-3, A-1

## C

- C language characters, 6-64
- C++ applications, order of recognition method values, 4-13
- Calendar user action command, 6-18
- CalendarVP verification point command, 6-20
- Call statement, 6-21
- CallScript utility command, 6-23
- CancelButton statement, 6-24
- Caption statement, 6-25
- caption terminator character
  - retrieving, 6-470
  - setting, 6-495
- caption wildcard characters, 4-17, 4-19
- case, 6-441
- case-sensitive comparison, 6-342
- CCur function, 6-26
- CDbl function, 6-27
- change directory, 6-27
- change drive, 6-28
- ChDir statement, 6-27
- ChDrive statement, 6-28
- CheckBox statement, 6-29
- CheckBox user action command, 6-30
- CheckBoxVP verification point command, 6-32
- child objects, 6-471
- child objects in recognition methods, 4-10, 4-18
  - Java commands, 4-13
- ChildWindow value, 4-19
- Chr function, 6-35
- ChrB, 6-35
- CInt function, 6-36



- Class List, 6-37
- Class property, 5-7
- clear Clipboard, 6-37
- clicking the mouse, E-1
- ClientRect property, 5-7
- Clipboard, 6-37
- ClipboardVP verification point command, 6-38
- CLng function, 6-39
- Close statement, 6-40
- combo box
  - elements in array, 6-111
  - fill with strings, 6-113
- ComboBox statement, 6-41
- ComboBox user action command, 6-42
- ComboBoxVP verification point command, 6-44
- ComboEditBox user action command, 6-48
- ComboEditBoxVP verification point command, 6-50
- ComboListBox user action command, 6-52
- ComboListBoxVP verification point command, 6-55
- Command function, 6-58
- commands in SQABasic. *See also* SQABasic command categories
  - additions to Basic, 1-3
  - functional listing of all commands, 2-1
  - types, 3-2
- comments, 6-421, A-2
- Comparators, displaying captured data, 5-12
- compare strings, 6-535
- comparing environment states, 5-21
- comparing Web sites, 6-579
- comparison operators, 3-13
- compiler directives command summary, 2-1
- compiling
  - library files, 4-27
  - scripts, 4-2
- concatenation operators, 3-12
- conditional execution, 6-217, 6-336, 6-441, 6-582
- console window
  - displaying messages in, 5-27
  - SQAConsoleWrite, 6-456
- Const statement, 6-59
- constants
  - global scope, 4-22
  - header files, 4-31
  - local scope, 4-21
  - module-level scope, 4-22
  - scope of, 3-14, 4-21
- contacting
  - technical publications, xxii
  - technical support, xxii
- context, 4-15
  - current, 6-586
  - default, 4-20
  - establishing, through a Window command action, 4-15
  - establishing, through context notation, 4-18
  - notation syntax, 4-18, A-2
  - Object Scripting commands and, 5-5
  - test, 4-16, 6-585, 6-587
  - window, 4-15, 4-16, 6-586
  - Window actions for setting, 4-16
- context window, 4-4
- convert to type
  - currency, 6-26
  - double, 6-27
  - general rules, 3-9
  - integer, 6-36
  - long, 6-39
  - single, 6-62
  - string, 6-63
  - variant, 6-66
  - variant date, 6-66
- Cos function, 6-60
- cosine, 6-60
- CreateObject function, 6-61
- creating dialog boxes, 6-10
- CSng function, 6-62
- CStr function, 6-63
- csv files, 5-12

## Index

- CurDir function, 6-65
- Currency data type, 3-6
- current
  - context window, 4-16, 6-586
  - date, 6-74, 6-75
  - directory, 6-65
- current baseline data files
  - copying to a logged baseline data file, 5-16
  - creating, 5-16
  - ownership, 5-20
  - retrieving the location, 6-505
  - vs logged baseline data files, 5-14
- CurrentFocus value, 5-5
- CurrentWindow, 6-588
- CurrentWindow value, 4-16, 5-5
- custom buttons, 6-16, 6-17
- custom code
  - header files, 4-29
  - library files, 4-26
  - scripts, 4-23
  - template file, 4-33
- custom procedures
  - adding to a library file, 4-26
  - adding to a script, 4-23
  - declaring in a header file, 4-29
  - declaring in a script, 4-23
- custom verification points
  - displaying captured data, 5-12
  - example, 5-17
  - managing, 5-12
  - retrieving actual file location, 6-503
  - retrieving baseline file location, 6-504
  - retrieving current baseline file location, 6-505
  - summary of management commands, 5-13
  - using, 5-16
  - writing results to the log, 6-506
- customer support, xxii
- customizing scripts, 4-20
- CVar function, 6-66
- CVDate function, 6-66

## D

- data types
  - converting, 3-9
  - declaring, 3-5, 6-97
  - default, 6-91
  - list of, 3-6
  - signed, 3-7
  - user-defined, 3-8, 6-570
- datapool commands, 1-3
  - overview, 5-30
  - summary, 2-2, 5-30
- datapools, role of, 5-30
- DataWindow user action command, 6-68
- DataWindowVP verification point command, 6-72
- date
  - day component, 6-80
  - format, 6-75, 6-160
  - is legal, 6-238
  - month component, 6-322
  - now, 6-330
  - value, 6-74, 6-76, 6-79
  - year component, 6-597
- date and time command summary, 2-2
- Date data types, 3-7, 3-8
- Date function, 6-74
- Date statement, 6-75
- DateSerial function, 6-76
- DateTime user action command, 6-77
- DateTimeVP verification point command, 6-78
- DateValue function, 6-79
- Day function, 6-80
- day of month, 6-80
- day of week, 6-581
- DDE, 5-31
  - command summary, 2-5
- DDEAppReturnCode function, 6-81
- DDEExecute statement, 6-82
- DDEInitiate function, 6-83
- DDEPoke statement, 6-85

- DDERequest function, 6-86
- DDETerminate statement, 6-88
- declaration statements summary, 2-2
- Declare statement, 6-89
- declaring
  - .dll files, 4-28
  - arrays, 3-10
  - data types, 6-97
  - SQABasic library files, 4-28
  - variables of a User-Defined data type, 3-8
- declaring procedures
  - in a header file, 4-29
  - in a script, 4-23
- declaring variables and constants
  - global scope, 4-22
  - header files, 4-29, 4-31
  - local scope, 4-21
  - module-level scope, 4-22
  - scope, 3-14
- default
  - context, 4-20
  - data type, 6-91
  - playback browser, 6-496
- Def $\textit{type}$  statement, 6-91
- DelayFor utility command, 6-92
- delete file, 6-284
- derived trigonometric functions, D-1
- Desktop user action command, 6-93
- dialog box definition command summary, 2-3
- dialog box services command summary, 2-4
- dialog boxes, 6-9
  - as windows, 4-15
  - begin/end, 6-9
  - captions, 6-25
  - closing, 6-107
  - commands for handling user actions, 2-4
  - creating, 6-10
  - declaring in instance of, 6-10
  - defining, 6-9, 6-10
  - displaying, 6-10, 6-94, 6-96
  - enable state, 6-104, 6-105
  - focus, 6-109, 6-110
  - handling user actions in, 6-10
  - numeric ID, 6-102
  - OptionGroup, 6-344
  - password box, 6-348
  - records, 3-9, 6-10
  - SQABasic, 6-9
  - text in, 6-548
- dialog controls
  - DropComboBox, 6-128
  - DropListBox, 6-129
  - hidden/visible, 6-123, 6-124
  - InputBox, 6-222
  - ListBox, 6-296
  - OK button, 6-335
  - OptionButton, 6-343
  - picture, 6-350
  - picture, 6-114
  - PushButton, 6-399
  - state, 6-120
  - StaticComboBox, 6-529
  - text, 6-116, 6-118
  - TextBox, 6-549
  - value, 6-122
- Dialog function, 6-94
- Dialog statement, 6-96
- Dim statement, 6-97
- dimension variables, 6-97
- dimensions of an array, 3-10, 6-98, 6-180
  - default lower bound, 6-340
  - lower bound, 6-289
  - upper bound, 6-573
  - with dynamic arrays, 3-11, 6-418
- Dir function, 6-100
- directory
  - attributes, 6-175
  - change, 6-27
  - contents, 6-100
  - create new, 6-319
  - log, 6-474
  - remove, 6-429
  - standard, 6-473
- disk and directory command summary, 2-4

## Index

### displaying

- custom verification point data, 5-12
- messages, 5-27
- DlgControlIID function, 6-102
- DlgEnable function, 6-104
- DlgEnable statement, 6-105, 6-107
- DlgFocus function, 6-109
- DlgFocus statement, 6-110
- DlgListBoxArray function, 6-111
- DlgListBoxArray statement, 6-113
- DlgSetPicture statement, 6-114
- DlgText function, 6-116
- DlgText statement, 6-118
- DlgValue function, 6-120
- DlgValue statement, 6-122
- DlgVisible function, 6-123
- DlgVisible statement, 6-124
- Do...Loop statement, 6-126
- document files, 6-501
- documentation feedback, xxii
- DoEvents statement, 6-127
- Double data type, 3-6
- dragging the mouse, E-1
- drive change, 6-28
- DropComboBox statement, 6-128
- DropListBox statement, 6-129
- dynamic arrays, 3-11, 6-418
- Dynamic Data Exchange, 5-31
  - close, 6-88
  - initiate, 6-83
  - receive data, 6-86
  - return code, 6-81
  - send commands, 6-82
  - send data, 6-85
- Dynamic Link Library
  - declare procedure, 6-89
  - library name, 6-89

## E

- EditBox user action command, 6-131
- EditBoxVP verification point command, 6-133

### editing scripts, 1-2

- elements
  - of arrays, 3-10
  - of User-Defined types, 3-8
- empty variant, 3-7
- end of file, 6-138
- EndSaveWindowPositions utility command, 6-136
- Environ function, 6-137
- Environment property, 5-7
- environment state, 5-21
  - comparison report, 6-467
  - summary of commands, 5-21
  - test overview, 5-22
- environmental control command summary, 2-5
- Eof function, 6-138
- Eqv logical operator, 3-14
- Erase statement, 6-139
- Erl function, 6-141
- Err function, 6-142
- Err statement, 6-143
- error codes, B-1
- Error function, 6-144
- error handling, 3-16
  - command summary, 2-5
  - halting, 6-423
  - location of routine, 6-337
  - message text, 6-144
  - Object Scripting commands, 5-11
  - runtime code, 6-143
  - script command failure, 6-494
  - trap line number, 6-141
  - trap runtime code, 6-142
  - user-defined, 6-145
- Error statement, 6-145
- escape character for wildcards, 4-17, 4-20
- execute query, 6-513
- Exit statement, 6-146
- Exp function, 6-147
- explicit data type declaration, 3-5
- exponent, 6-147
- expressions, 3-12
  - Null, 6-240, 6-241

## F

- factorials, 6-62, 6-155
- feedback, xxii
- field names and PeopleTools object names, 6-381, 6-385
- fields replaced with strings, 6-446
- file control command summary, 2-6
- FileAttr function, 6-148
- FileCopy statement, 6-149
- FileDateTime function, 6-150
- FileLen function, 6-151
- files
  - actual data, 4-7, 5-12, 5-15, 5-16
  - attributes, 6-175
  - baseline, 4-7, 5-16
  - close all, 6-422
  - closing, 6-40
  - copying, 6-149
  - current baseline, 5-16
  - current offset, 6-307
  - date and time of, 6-150
  - deleting, 6-284
  - end of, 6-138
  - header, 4-29
  - in a directory, 6-100
  - included, 4-29, 6-218
  - input from, 6-220, 6-221, 6-294
  - length of, 6-151, 6-310
  - link, 6-501
  - locking, 6-308
  - logged baseline, 5-16
  - low-level journal, 6-351
  - moving, 6-326
  - opening, 6-338
  - output width, 6-584
  - printing to, 6-356
  - read data, 6-173
  - renaming, 6-326
  - reset, 6-422
  - seek position, 6-438, 6-439
  - set attributes, 6-444
  - summary of input/output commands, 2-6
  - system handle, 6-148
  - types of, 6-501
  - unlocking, 6-575
  - unused number, 6-164
  - writing data, 6-405, 6-595
- FileVP verification point command, 6-152
- financial
  - constant periodic payment, 6-352
  - function summary, 2-7
  - interest payment, 6-234
  - interest per period, 6-413
  - net present value, 6-330
  - present value, 6-406
  - principal amount, 6-355
  - rate of return, 6-236
- Fix function, 6-153
- fixed-length strings, 3-6
- flow control statement summary, 2-7
- For...Next statement, 6-154
- Format function, 6-156
- formatting
  - date and time, 6-160
  - numbers, 6-157
  - strings, 6-163
- four-digit years, 3-15
- FreeFile function, 6-164
- FullRecognition property, 5-7
- function procedures. *See* functions
- Function...End Function statement, 6-164
- functional listing of commands, 2-1
- functional testing, 1-4
- functions
  - adding to a library file, 4-26
  - adding to a script, 4-23
  - calling, 6-21
  - custom, 4-20
  - declaration syntax, A-2
  - declaring in a header file, 4-29
  - declaring in a script, 4-23, 4-24

## Index

- defining, 6-164
- description of, 3-2
- global scope, 4-26
- module-level scope, 4-23

future value, 6-166  
FV function, 6-166

## G

GenericObject user action command, 6-168  
GenericObjectVP verification point command, 6-170  
get schema, 6-514  
Get statement, 6-173  
GetAttr function, 6-175  
GetField function, 6-176  
GetLastVPResult utility command, 6-177  
GetObject function, 6-177  
global scope

- constants, 4-22, 6-59
- header files, 4-31
- procedures, 4-26
- variables, 4-22, 6-179

Global statement, 6-179  
global.sbh, 4-30  
global.sbl, 4-27  
GoTo statement, 6-182  
GroupBox statement, 6-183  
GroupBox user action command, 6-184  
GroupBoxVP verification point command, 6-186

## H

halt execution, 6-534  
header files, 4-26, 4-29

- declarations in, 4-31
- referencing, 4-31
- scope, 4-31

Header user action command, 6-188  
HeaderVP verification point command, 6-190  
help desk, xxii  
Hex function, 6-192

hierarchical objects in recognition methods, 4-10, 4-18  
HotKeyControl user action command, 6-193  
HotKeyControlVP verification point command, 6-194  
hotline support, xxii  
Hour function, 6-195  
HTML user action command, 6-196  
HTMLActiveX user action command, 6-200  
HTMLActiveX VP verification point command, 6-201  
HTMLDocument user action command, 6-203  
HTMLDocumentVP verification point command, 6-204  
HTMLHiddenVP verification point command, 6-206  
HTMLImage user action command, 6-208  
HTMLImageVP verification point command, 6-209  
HTMLink user action command, 6-211  
HTMLinkVP verification point command, 6-212  
HTMLTable user action command, 6-214  
HTMLTableVP verification point command, 6-215  
HTMLVP verification point command, 6-198  
HTTP requests, 1-4  
hWnd property, 5-7

## I

If...Then...Else, 6-217  
Imp logical operator, 3-14  
implicit data type declaration, 3-5  
inactive window, 6-586  
Include files, 4-26, 4-31

- adding to the template, 4-33

including files, 4-29, 6-218  
initializing scripts, 4-2  
Input # statement, 6-221  
input boxes, 6-222  
Input function, 6-220  
InputB, 6-220  
InputBox function, 6-222  
InputChars user action command, 6-223  
InputKeys user action command, 6-224

Installation Analyzer utility, 5-21  
 instance of a dialog box, 3-9  
 InStr function, 6-229  
 InStrB, 6-229  
 Int function, 6-230  
 Integer data type, 3-6  
 IPAddress user action command, 6-232  
 IPAddressVP verification point command, 6-233  
 IPmt function, 6-234  
 IRR function, 6-236  
 Is Operator, 6-237  
 IsDate function, 6-238  
 IsEmpty function, 6-238  
 IsMissing function, 6-239  
 IsNull function, 6-240  
 IsNumeric function, 6-241

**J**

Java applications, starting, 6-524  
 Java commands and recognition methods, 4-13  
 Java objects and Object Scripting commands, 4-14  
 JavaCanvas user action command, 6-242  
 JavaCanvasVP verification point command, 6-244  
 JavaListView user action command, 6-246  
 JavaListViewVP verification point command, 6-248  
 JavaMenu user action command, 6-250  
 JavaMenuVP verification point command, 6-251  
 JavaObject user action command, 6-253  
 JavaObjectVP verification point command, 6-255  
 JavaPanel user action command, 6-256  
 JavaPanelVP verification point command, 6-258  
 JavaPopupMenu user action command, 6-260  
 JavaPopupMenuVP verification point command,  
 6-261  
 JavaSplitPane user action command, 6-263  
 JavaSplitPaneVP verification point command, 6-264  
 JavaSplitter user action command, 6-266  
 JavaSplitterVP verification point command, 6-268  
 jTable user action command, 6-270  
 jTableHeader user action command, 6-273

jTableHeaderVP verification point command, 6-275  
 jTableVP verification point command, 6-272  
 JavaTree user action command, 6-277  
 JavaTreeVP verification point command, 6-279  
 JavaWindow user action command, 6-280  
 JavaWindowVP verification point command, 6-282

**K**

keyboard input, 6-224  
 keystrokes
 

- InputChars, 6-223
- InputKeys, 6-224

 Kill statement, 6-284

**L**

Label user action command, 6-285  
 labels in SQABasic code, A-3  
 LabelVP verification point command, 6-286  
 language elements, 3-1  
 LBound function, 6-289  
 LCase function, 6-290  
 Left function, 6-290  
 LeftB, 6-291  
 Len function, 6-292  
 LenB, 6-292  
 Let statement, 6-292  
 library file location
 

- .dll, 4-29
- SQABasic library (.rec), 4-26
- SQABasic library (.sbx), 4-26

 library files
 

- compiling, 4-27
- creating, 4-27
- declaring (.dll), 4-28
- declaring (SQABasic .rec), 4-28
- declaring (SQABasic .sbx), 4-28
- including, 4-29, 6-218

 library names and Dynamic Link Libraries, 6-89  
 Like Operator, 6-293

## Index

- line continuation syntax, A-3
- Line Input statement, 6-294
- line numbers not supported, A-3
- link files, 6-501
- links, testing for problems with, 6-579
- list box
  - elements in array, 6-111
  - fill with strings, 6-113
- ListBox Statement, 6-296
- ListBox user action command, 6-297
- ListBoxVP verification point command, 6-300
- ListView user action command, 6-303
- ListViewVP verification point command, 6-305
- Loc function, 6-307
- local scope
  - constants, 4-21
  - variables, 4-21
- Lock statement, 6-308
- Lof function, 6-310
- Log function, 6-311
- log messages
  - results of user-defined test, 6-597
  - SQALogMessage, 5-28
  - SQAScriptCmdFailure, 5-17, 5-29
  - writing, 5-28, 6-492
- log path, finding, 6-474
- logged baseline data files
  - copying from a current baseline data file, 5-16
  - ownership, 5-20
  - vs current baseline data files, 5-14
- logical operators, 3-13
- LogViewer
  - displaying captured data, 5-12
  - displaying messages in, 5-28
  - SQALogMessage, 6-492
  - SQAScriptCmdFailure, 6-494
  - SQAVpLog, 6-506
- Long data type, 3-6
- loops
  - Do/While, 6-126
  - exiting, 6-146
  - For/Next, 6-154

- lower bound, 6-289
  - default, 6-340
- lower case, 6-290
- low-level files, 6-351
- Lset statement, 6-312
- LTrim function, 6-313

## M

- managing custom verification points, 5-12
- MDI windows, 4-19
- menu items
  - MenuIDSelect, 6-314
  - MenuSelect, 6-314
  - PopupMenuIDSelect, 6-353
  - PopupMenuSelect, 6-353
  - SysMenuIDSelect, 6-539
  - SysMenuSelect, 6-540
- MenuIDSelect user action command, 6-314
- MenuSelect user action command, 6-314
- message boxes, 6-322, 6-325
- messages
  - console window, 5-27
  - LogViewer, 5-28
  - overview, 5-26
  - results of user-defined tests, 6-597
  - SQAConsoleWrite, 6-456
  - SQALogMessage, 6-492
  - SQAScriptCmdFailure, 6-494
  - SQAVpLog, 6-506
- metacommands
  - 'SCStrings, 6-64
  - 'SInclude, 4-29, 6-218
  - '\$NoCStrings, 6-328
- methods, 5-32
  - Clipboard, 6-37
  - execute an object's methods, 6-490
- Mid function, 6-316
- Mid statement, 6-317
- MidB, 6-316, 6-318
- Minute function, 6-318



missing arguments, 6-239  
 missing links, testing for, 6-579  
 Mkdir statement, 6-319  
 ModuleFileName property, 5-7  
 module-level scope  
   constants, 4-22  
   procedures, 4-23  
   variables, 4-22  
 modules, 4-22  
 ModuleVP verification point command, 6-320  
 Month function, 6-322  
 mouse actions, E-1  
 moving files, 6-326  
 MsgBox function, 6-322  
 MsgBox statement, 6-325

## N

name format in SQABasic, A-3  
 Name property, 5-8  
 Name statement, 6-326  
 named arguments, 3-4, 6-22  
 nested scripts, 6-23  
 net present value, 6-330  
 new directory, 6-319  
 New Operator, 6-327  
 Not logical operator, 3-13  
 Nothing function, 6-329  
 Now function, 6-330  
 NPV function, 6-330  
 null  
   expression, 6-240  
   variables, 6-331  
   variant, 3-8  
 Null function, 6-331  
 numbers  
   absolute value, 6-2  
   as string, 6-535  
   formatted, 6-157  
   global, 6-180  
   hexadecimal, 6-192  
   integer, 6-230  
   integer part, 6-153  
   logarithm, 6-311  
   octal, 6-334  
   random, 6-412, 6-430  
   sign of, 6-448  
   square root, 6-521  
   value in string, 6-577  
 numeric  
   operators, 3-12  
   variables, 6-98  
 numeric function summary, 2-8

## O

Object Class, 6-332  
 object command summary, 2-9  
 object context. *See* context  
 Object data type, 3-6  
 object handling, 5-32  
 Object Properties verification point, 5-6  
 Object Scripting commands, 1-3, 5-1  
   Java objects, 4-14  
   object context, 5-5  
   object types and, 5-2  
   specifying an object, 5-2  
   specifying an object property, 5-6  
   status codes for, 5-11  
   status codes for (list), C-1  
   summary, 2-8  
   types of properties to access, 5-6, 5-7  
 object variables, 6-98  
 objects  
   child, 6-471  
   class type, 6-572  
   compare, 6-237  
   current focus, 5-5  
   currently active window, 4-16, 5-5  
   data type, 3-6  
   getting a property value for, 6-478, 6-485  
   getting an array of a property's values, 6-480, 6-482  
   hierarchical order, in context notation, 4-18  
   retrieving property names for, 6-487

## Index

- searching for, 6-469
  - setting property values for, 6-498
  - specifying, 5-2
  - SQABasic names for, 5-3
  - types of, 5-3
  - waiting for appearance of, 6-508
- ObjectType property, 5-8
- OCR region
- coordinates, 6-474
  - text, 6-476
- Oct function, 6-334
- ODBC
- close source, 6-511
  - errors, 6-512
  - function summary, 2-9
  - open source, 6-516
- OKButton statement, 6-335
- OLE2, 5-32
- assign variable, 6-443
  - associated object, 6-177
  - automation object, 6-61
  - new object, 6-327
  - object class, 6-332
- On Error statement, 6-337
- On...GoTo statement, 6-336
- Open statement, 6-338
- opening files, 6-501
- operating system, determining the type, 6-488
- operators, 3-12
- comparison, 3-13
  - logical, 3-13
  - numeric, 3-12
  - string concatenation, 3-12
- Option Base statement, 6-340
- Option Compare statement, 6-342
- Option Explicit statement, 6-100, 6-343
- OptionButton statement, 6-343
- Or logical operator, 3-13
- order of recognition method values, 4-10
- changing, 4-12
- ordinal, 6-89
- output width, 6-584
- ownership of custom verification point files, 5-20

## P

- Pager user action command, 6-346
- PagerVP verification point command, 6-347
- panel objects on PeopleTools panels, 6-381, 6-385
- parameters for user actions, 4-9, A-3
- parameters\$ argument, 4-9, A-3
- parent objects in recognition methods, 4-10, 4-18
- Java commands, 4-13
- ParentRecognition property, 5-8
- passing arguments, 3-3, 6-22
- PasswordBox function, 6-348
- pattern matching, 6-293
- pause script execution, 6-92
- PeopleTools panel object names, 6-381
- performance testing, 1-4
- Picture statement, 6-350
- pictures in dialog controls, 6-114
- pipe separator ( | ), 6-364
- playback of a verification point, 4-7
- custom verification points, 5-16
- PlayJrnl utility command, 6-351
- Pmt function, 6-352
- pointer separator ( -> )
- PeopleSoft derived fields, 6-381, 6-385
  - PSTree, 6-392
  - PSTreeVP, 6-398
  - TreeView, 6-565
  - TreeViewVP, 6-569
- popup menus, 6-353
- PopupMenuIDSelect user action command, 6-353
- PopupMenuSelect user action command, 6-353
- PPmt function, 6-355
- Print statement, 6-356

- procedures
    - adding to a library file, 4-26
    - adding to a script, 4-23
    - declaring in a header file, 4-29
    - declaring in a script, 4-23
    - global scope, 4-26
    - module-level scope, 4-23
  - ProgressBar user action command, 6-358
  - ProgressBarVP verification point command, 6-359
  - project header files, 4-29
  - projects, location, 6-473
  - properties, 5-32
    - additional, with Object Scripting commands, 5-7
    - retrieving a value, 6-478, 6-485
    - retrieving an array of values, 6-480, 6-482
    - retrieving the names of, 6-487
    - retrieving the number of elements in an array, 6-483
    - setting a value for, 6-498
    - specifying, 5-6
    - types you can access, 5-6
    - waiting for a particular value, 6-509
  - PSGrid user action command, 6-362
  - PSGridHeader user action command, 6-365
  - PSGridHeaderVP verification point command, 6-366
  - PSGridVP verification point command, 6-368
  - PSMenu user action command, 6-371
  - PSMenuVP verification point command, 6-372
  - PSNavigator user action command, 6-373
  - PSNavigatorVP verification point command, 6-375
  - PSPanel user action command, 6-378
  - PSPanelVP verification point command, 6-382
  - PSSpin user action command, 6-386
  - PSSpinVP verification point command, 6-388
  - PSTree user action command, 6-390
  - PSTreeHeader user action command, 6-392
  - PSTreeHeaderVP verification point command, 6-394
  - PSTreeVP verification point command, 6-396
  - PushButton statement, 6-399
  - PushButton user action command, 6-400
  - PushButtonVP verification point command, 6-402
  - Put statement, 6-405
  - PV function, 6-406
- ## Q
- qualifiers in recognition methods, 4-10
- ## R
- radians, 6-8
  - RadioButton user action command, 6-407
  - RadioButtonVP verification point command, 6-409
  - Randomize statement, 6-412
  - Rate function, 6-413
  - Rational ActiveX Test Control, 6-523
  - Rational technical publications, contacting, xxii
  - Rational technical support, xxii
  - Rebar user action command, 6-414
  - RebarVP verification point command, 6-416
  - rec files, 1-3, 4-2
    - as library files, 4-26
  - recMethod\$ argument, 4-8, 5-2, A-4
    - context notation and, 4-18
    - getting help defining, 5-9
    - Java commands and, 4-13
    - multiple values in, 4-10
  - recognition method. *See* recMethod\$ argument
  - recognition methods
    - changing the default order, 4-12
    - context notation and, 4-18
    - getting help defining, 5-9
    - Java commands and, 4-13
    - multiple values in, 4-10
    - order of values, 4-10
    - overview, 4-8
    - specifying when the object name is unknown, 4-16, 5-5
  - Recognition property, 5-8

## Index

Record data type. *See* User-Defined data type  
record names and PeopleTools object names, 6-381, 6-385  
record of a dialog box, 3-9, 6-10  
recorded baseline, 1-3, 4-7  
    custom verification points, 5-12  
recording a verification point, 4-7  
    custom verification points, 5-16  
rectangle of an OCR region, 6-474  
ReDim statement, 6-417  
referencing library files, 4-29, 6-218  
RegionVP verification point command, 6-419  
Registry changes, testing for, 5-21  
Rem statement, 6-421  
renaming files, 6-326  
repeated character strings, 6-537  
repetitive action, 6-582  
repository, location, 6-473  
repository-wide header files, 4-29  
request data, 6-517  
Reset statement, 6-422  
ResetTime utility command, 6-423  
restoring windows during playback, 4-3  
Resume statement, 6-423  
retrieve data, 6-518  
    to file, 6-520  
return ASCII value, 6-7  
RichEdit user action command, 6-424  
RichEditVP verification point command, 6-426  
Right function, 6-428  
RightB, 6-428  
Rmdir statement, 6-429  
Rnd function, 6-430  
Rset statement, 6-431  
RTrim function, 6-432  
Run Now check box, 6-23  
runtime  
    errors, 3-16  
    file location, 6-473  
    files, 1-3

## S

saving  
    header files, 4-30  
    library files, 4-27  
    scripts, 4-2  
sbh files, 1-3, 4-29  
sbl files, 1-3, 4-26  
sbx files, 1-3, 4-2  
scope  
    global, 4-22  
    header files, 4-29, 4-31  
    procedures, 4-23, 4-26  
    variables and constants, 3-14, 4-21  
screen I/O command summary, 2-10  
ScreenRect property, 5-8  
scripts  
    automatic generation of, 1-1  
    body of, 4-4  
    calling from another script, 6-23  
    compiling, 4-2  
    customizing, 4-20  
    declarations in, 4-21, 4-23  
    ending, 4-4  
    example, 4-4  
    initializing, 4-2  
    location, 6-473  
    nested, 6-23  
    overview, 4-1  
    pausing execution of, 6-92  
    reasons for editing, 1-2  
    saving, 4-2  
    sections of, 4-2  
    syntax summary, A-1  
    template file, 4-33  
    window restoration section, 4-3  
ScrollBar user action command, 6-433  
ScrollBarVP verification point command, 6-435  
Second function, 6-437  
Seek function, 6-438

- Seek statement, 6-439
- Select Case statement, 6-441
- select menu items
  - MenuIDSelect, 6-314
  - MenuSelect, 6-314
  - PopupMenuIDSelect, 6-353
  - PopupMenuSelect, 6-353
  - SysMenuIDSelect, 6-539
  - SysMenuSelect, 6-540
- SendKeys statement, 6-443
- separator character, 6-500
- Set statement, 6-443
- set system date, 6-75
- set system time, 6-551
- SetAttr statement, 6-444
- SetField function, 6-446
- SetThinkAvg timing and coordination command, 6-447
- SetTime utility command, 6-448
- Sgn function, 6-448
- Shell function, 6-449
- shortcut files, 6-501
- signed data types, 3-7
- Sin function, 6-450
- sine, 6-450
- Single data type, 3-6
- socket-level requests, 1-4
- source files, 1-3
  - library, 4-26
  - scripts, 4-2
- Space function, 6-451
- spaces
  - printing, 6-452
  - string of, 6-451
- Spc function, 6-452
- special characters, 6-64
- SpinControl user action command, 6-453
- SpinControlVP verification point command, 6-454
- SQABasic
  - .rec files, 4-2, 4-26
  - .sbh files, 4-29
  - .sbl files, 4-26
  - .sbx files, 4-2, 4-27
  - .tpl files, 4-33
  - access to external objects, 5-31
  - additions to Basic commands, 1-1, 1-3, 6-1
  - commands, 3-2
  - context notation, 4-18
  - custom code, 4-23, 4-26
  - custom functions, 4-20
  - custom sub procedures, 4-20
  - dialog boxes, 6-9
  - error handling, 3-16
  - files, 1-3
  - header files, 4-29
  - language elements, 3-1
  - library files, 4-26
  - name format, A-3
  - object handling, 5-32
  - syntax summary, A-1
  - template file, 4-33
  - unique commands, 1-4
- SQABasic command categories, 2-10
  - datapool, 2-2
  - Object Scripting, 2-8
  - overview, 1-3
  - timing and coordination, 2-12
  - user action, 2-12
  - utility, 2-14
  - verification point, 2-17
- SQAConsoleClear utility command, 6-456
- SQAConsoleWrite utility command, 6-456
- SQADatapoolClose datapool command, 6-457
- SQADatapoolFetch datapool command, 6-458
- SQADatapoolOpen datapool command, 6-459
- SQADatapoolRewind datapool command, 6-462
- SQADatapoolValue datapool command, 6-463
- SQAEnvCreateBaseline utility command, 6-465
- SQAEnvCreateCurrent utility command, 6-466
- SQAEnvCreateDelta utility command, 6-467
- SQAFindObject Object Scripting command, 6-469
- SQAGetCaptionTerminatorChar utility command, 6-470
- SQAGetChildren Object Scripting command, 6-471

## Index

- SQAGetDir utility command, 6-473
- SQAGetLogDir utility command, 6-474
- SQAGetOcrRegionRect utility command, 6-474
- SQAGetOcrRegionText utility command, 6-476
- SQAGetProperty Object Scripting command, 6-478
- SQAGetPropertyArray Object Scripting command, 6-480
- SQAGetPropertyArrayAsString Object Scripting command, 6-482
- SQAGetPropertyArraySize, 6-483
- SQAGetPropertyAsString Object Scripting command, 6-485
- SQAGetPropertyNames Object Scripting command, 6-487
- SQAGetSystemLong utility command, 6-488
- SQAINvokeMethod Object Scripting command, 6-490
- SQALogMessage utility command, 6-492
- SQAQueryKey utility command, 6-493
- SQARectangle User-Defined data type, 6-475
- SQAResumeLogOutput Utility Command, 6-493
- SQAScriptCmdFailure utility command, 6-494
- SQASetAssignmentChar utility command, 6-494
- SQASetCaptionTerminatorChar utility command, 6-495
- SQASetDefaultBrowser utility command, 6-496
- SQASetProperty Object Scripting command, 6-498
- SQASetSeparatorChar utility command, 6-500
- SQAShellExecute utility command, 6-501
- SQASuspendLogOutput utility command, 6-502
- SQASyncPointWait timing and coordination command, 6-502
- SQAVpGetActualFileName utility command, 6-503
- SQAVpGetBaselineFileName utility command, 6-504
- SQAVpGetCurrentBaselineFileName utility command, 6-505
- SQAVpLog utility command, 6-506
- SQAWaitForObject Object Scripting command, 6-508
- SQAWaitForPropertyValue, 6-509
- SQL requests, 1-4
- SQLClose function, 6-511
- SQLError function, 6-512
- SQLExecQuery function, 6-513
- SQLGetSchema function, 6-514
- SQLOpen function, 6-516
- SQLRequest function, 6-517
- SQLRetrieve function, 6-518
- SQLRetrieveToFile function, 6-520
- Sqr function, 6-521
- square root, 6-521
- start application
  - Shell command, 6-449
  - SQAShellExecute command, 6-501
  - StartApplication command, 6-522
  - StartJavaApplication command, 6-524
- StartApplication utility command, 6-522
- StartBrowser utility command, 6-523
- starting a browser, 6-523
- starting a timer, 6-527
- StartJavaApplication utility command, 6-524
- StartSaveWindowPositions utility command, 6-526
- StartTimer utility command, 6-527
- statements, 3-2
- Static statement, 6-527
- StaticComboBox statement, 6-529
- status codes for Object Scripting commands, 5-11
  - list, C-1
- StatusBar user action command, 6-530
- StatusBarVP verification point command, 6-532
- Stop statement, 6-534
- stopping a timer, 6-534
- StopTimer utility command, 6-534
- Str function, 6-535
- StrComp function, 6-535
- String function, 6-537
- string variable syntax, A-4
- strings
  - comparing, 6-342
  - concatenation operator, 3-12
  - converting to lower case, 6-290
  - converting to upper case, 6-575
  - copying, 6-312
  - data type, 3-6

- finding substrings in, 6-316, 6-317, 6-428
- fixed-length, 3-6, 6-99
- global, 6-181
- pattern matching, 6-293
- right align, 6-431
- summary of conversion functions, 2-10
- summary of manipulation functions, 2-11
- trimming spaces, 6-313, 6-432, 6-570
- types of, 6-99
- variable length, 3-6, 6-99
- sub procedures
  - adding to a library file, 4-26
  - adding to a script, 4-23
  - calling, 6-21
  - custom, 4-20
  - declaration syntax, A-4
  - declaring in a header file, 4-29
  - declaring in a script, 4-23
  - defining, 6-538
  - description of, 3-2
  - global scope, 4-26
  - module-level scope, 4-23
- sub programs. *See* sub procedures
- Sub...End Sub statement, 6-538
- subscripts of an array, 3-10, 6-98, 6-180
  - default lower bound, 6-340
  - lower bound, 6-289
  - omitted with dynamic array declarations, 3-11
  - upper bound, 6-573
- support, technical, xxii
- suspend log output, 6-502
- symbolic constants, 6-59
- syntax of user action and verification point
  - commands, 4-8
- syntax summary, A-1
- SysMenuIDSelect user action command, 6-539
- SysMenuSelect user action command, 6-540
- system environment, 6-137
- system events, 6-127
  - retrieve value, 6-488

- System menu select
  - by ID, 6-539
  - by text, 6-540

## T

- Tab function, 6-541
- TabControl user action command, 6-542
- TabControlVP verification point command, 6-544
- Tan function, 6-547
- tangent, 6-547
- technical support, xxii
- template file, 4-33
- test context, 4-16, 6-585, 6-587
- Text statement, 6-548
- text to/from Clipboard, 6-37
- TextBox statement, 6-549
- think time, 6-447
- time
  - as value, 6-554
  - as variant, 6-553
  - current, 6-550
  - format, 6-160
  - hour of day, 6-195
  - minute component, 6-318
  - now, 6-330
  - reset delay, 6-423
  - seconds component, 6-437
  - set average think, 6-447
  - set delay, 6-448
  - set system, 6-551
- Time function, 6-550
- Time statement, 6-551
- Timer function, 6-552
- timers
  - starting, 6-527
  - stopping, 6-534
  - system, 6-552
- TimeSerial function, 6-553
- TimeValue function, 6-554

## Index

- timing and coordination commands, 1-3
  - summary, 2-12
- title bar wildcard characters, 4-17, 4-19
- Toolbar user action command, 6-555
- ToolbarVP verification point command, 6-556
- Trackbar user action command, 6-559
- TrackbarVP verification point command, 6-561
- trappable errors, B-1
- trapping errors, 3-16
  - line number, 6-141
  - message text, 6-144
  - runtime code, 6-142, 6-143
  - user-defined, 6-145
- TreeView user action command, 6-563
- TreeViewVP verification point command, 6-567
- trigonometric function summary, 2-8
- Trim function, 6-570
- TUXEDO, 1-4
- two-digit years, 3-15
- Type statement, 6-570
- type-declaration characters, 3-5
- Typeof function, 6-572
- types of
  - objects, 5-3
  - properties, 5-6
- TypingDelays timing and coordination command, 6-572

## U

- UBound function, 6-573
- UCase function, 6-575
- unique SQABasic commands, 1-4
- Unlock statement, 6-575
- upper bound, 6-573
- upper case, 6-575
- user action commands, 1-3
  - arguments in, 4-8
  - overview, 4-6
  - summary, 2-12
  - syntax, 4-8

- user actions, 4-4, 4-8
  - context for, 4-15
- User-Defined data type, 3-6, 3-8, 6-99
  - declaring, 3-8, 6-99
  - defining, 6-570
  - global, 6-180
  - reassigning to another variable, 6-312
  - referencing, A-4
- user-defined errors, 3-17
- utility commands, 1-3
  - summary, 2-14

## V

- Val function, 6-577
- value
  - absolute, 6-2
  - ASCII, 6-7
  - constant, 6-59
  - date, 6-74, 6-76, 6-79
- variable-length strings, 3-6
- variables
  - arrays, 6-98
  - assignment, 6-292
  - declare type, 6-97, 6-343
  - define default type, 6-91
  - empty, 6-99
  - global scope, 4-22, 6-179
  - header files, 4-31
  - length of, 6-292
  - local scope, 4-21
  - module-level scope, 4-22
  - name format, A-3
  - Null, 6-331
  - numeric, 6-98
  - object, 6-98
  - scope of, 3-14, 4-21
  - static, 6-527
  - string, 6-99
  - user-defined, 6-99, 6-570
  - variant, 6-99



Variant data type, 3-6  
   declaring, 6-99  
   empty, 6-238  
   explicit and implicit declaration, 3-7  
   global, 6-181  
   identifying the type of data stored, 3-8, 6-578  
   initialized, 6-238  
 Variants command summary, 2-17  
 VarType function, 6-578  
 verification point data files  
   retrieving actual file location, 6-503  
   retrieving baseline file location, 6-504  
   retrieving current baseline file location, 6-505  
 verification points  
   baseline, 4-7  
   command summary, 2-17  
   command syntax, 4-8  
   commands, 1-3  
   comparing baseline and actual data, 4-7  
   custom. *See* custom verification points  
   in library files, 4-26  
   in scripts, 4-4  
   location, 6-473  
   overview, 4-7  
   ownership, 5-20  
   pass or fail, 1-3  
 version of Robot, determining, 6-488

## W

Web  
   default playback browser, 6-496  
   starting a browser, 6-523  
   testing a site for defects, 6-579  
 WebSiteVP verification point command, 6-579  
 Weekday function, 6-581  
 While...Wend, 6-582

Width statement, 6-584  
 wildcards for window captions, 4-17, 4-19  
 window  
   activate, 6-6  
   caption terminator character, 6-495  
   context, 4-4  
   making active or keeping inactive, 6-586  
   property for window handle, 5-7  
   save position, 6-136, 6-526  
   wildcards in captions, 4-17, 4-19  
 Window user action command, 6-585  
 windows  
   child window, 4-19  
   context for actions, 4-15  
   definition of, 4-15  
   MDI, 4-19  
   restoring, 4-3  
 Windows operating system, determining the type,  
   6-488  
 WindowVP verification point command, 6-591  
 With statement, 6-594  
 Write statement, 6-595  
 writing  
   console window, 5-28  
   LogViewer, 5-28  
   to a file, 6-405

## X

Xor logical operator, 3-14

## Y

year 2000 considerations, 3-15  
 year formats, 3-15  
 Year function, 6-597

Index