RSIScript Reference Manual

A Scripting Language for RSIGuard v6

Table of Contents

RSIScript Overview

RSIScript is a powerful scripting language that can be used to control RSIGuard and perform simple user-interface functions related to RSIGuard. For security reasons, commands with an * can only be executed as part of configuration scripts stored in the folder that contains RSIGuard.exe.

This document explains how to use RSIScript, and provides detailed documentation of the scripting language and its commands. It also describes how to create RSIGuard scripts to globally and semi-globally manage RSIGuard user configuration, send users messages, track RSIGuard usage. Additionally, it describes how to create macros and hotkey-triggered macros to perform useful functions in order to reduce keyboard and mouse-related strain.

RSIScript is a living language, and the available commands are updated regularly. Some commands may not exist in your "older version" of RSIGuard. The "NoErr" command can be used for backwards compatibility to prevent errors if your script will be used by people with various versions of RSIGuard.

RSIScript Security Considerations

RSIScript supports 2 levels of privilege.

Privileged scripts: Built in scripts in the "C:\Program Files\RSIGuard" folder are executed as privileged scripts and can use all commands available in RSIScript. Hotkey's that execute RSIScript also run in privileged mode.

Non-privileged scripts: All other script files run in non-privileged mode. Commands in this document that are preceded by an '*' are considered "restricted" commands. To use restricted commands in non-privileged script files, you must precede their usage with a "validate" command. The "validate" command asks the user for permission to execute a script with restricted commands. Some unrestricted commands have limitations if they are not running in a privileged script. If a script launches another script (e.g. via the 'read' command) or includes commands with embedded RSIScript (e.g. the 'question' and 'html' commands), all RSIScript commands will run at the same privilege level as the parent script. The 'html' command can only open URLs in the rsiguard.com and remedyinteractive.com domain unless they are running in privileged mode. Links in 'html' windows that use the "rsiguard:" protocol to execute RSIScript will execute the script with the same privilege as the script that executed the initial 'html' command. The 'msg log' command require privileged mode. In non-privileged mode, the 'menuedit add' command adds menu items that will always run the associated RSIScript commands in non-privileged mode. In environments in which RSIGuard communicates with Remedy Interactive's OES servers, all scripts delivered from the OES server run in non-privileged mode, and these scripts cannot be elevated to privileged mode via the 'validate' command. Commands with '**' can only be executed in scripts that are delivered and signed by the OES.

What Goes in an RSIScript File?

An RSIScript file should be created as you would create any other text file (e.g. with a text editor like Windows Notepad). Editors like Microsoft Word or Wordpad can be used, but when saving the file, you must specify that you want plain text format. RSIGuard defines the file extension RXS to be an RSIScript file, although TXT works fine too

The syntax (i.e. format) of an RSIScript file is as follows:

Each line in an RSIScript file can either: be blank; contain a command; or contain a comment (a documentation message). The following 3-line file, for example, sets the organization name. The first line is a comment because it starts with '#'. It is only there to remind you of what this script does. The third line actually sets the organization name.

```
# This script will set our organization's name within RSIGuard

set m\RSIGuard\OrganizationName ACME, Inc.
```

Command lines must begin with a command, and be followed by any options (also known as arguments) that are appropriate for the command.

Special RSIScript Files

During the course of normal operation, RSIGuard will try to run certain special scripts, if you have created them, to allow you to control and customize RSIGuard operation. These are analogous to DOS scripts like Config.sys and Autoexec.bat or Unix scripts like .login, .cshrc, etc. Special RSIGuard scripts can be created that:

- Run when a copy of RSIGuard is registered (i.e. one-time actions you want to occur on each RSIGuard computer). RSIGuard Software staff will generally create this "registration" script file that sets the default machine-based settings you request (e.g. organization name), and otherwise customizes RSIGuard to the needs of your organization. This script is generally not located on the user's computer – rather it is transmitted when the registration code is entered from the RSIGuard website.
- **Register.txt:** Run right after RSIGuard is registered. If the file Register.txt exists in the installation folder, Register.txt will be executed after registration (in addition to any script returned by use of a web-based registration code (i.e. one that starts with 'x')).
- **InstRun.txt:** Run the first time RSIGuard ever runs on a computer (uninstalling and reinstalling will NOT cause this script to run again because the first run is flagged in a registry value). This script can do initial configuration tasks that do not require an RSIGuard profile (e.g. set u\$p* is not allowed at this point). This script should be in a file InstRun.txt stored in the RSIGuard executable folder.
- **FirstRun.txt:** Run the first time a new version of RSIGuard is run for the first time. This script can do initial configuration tasks that do not require an RSIGuard profile (e.g. set u\$p* is not allowed at this point). This script should be in a file FirstRun.txt stored in the RSIGuard executable folder. This script is run immediately after InstRun.txt if this is the first run after installation. This file is often included in an MSI package for registering RSIGuard or for establishing a connection to a Remedy Interactive OES database.
- **Newuser.txt:** Run each time a new user or profile is created (i.e. actions you want to occur separately for each user and each users's various profiles). This script can be used to do things like set a default break frequency. This script is by default stored in the installation folder under the name newuser.txt, but you can change that (e.g. in the registration script with a command like "set m\RSIGuard\NewUserScript http://yourdomain.com/newusr.txt")
- **Newuseronline.txt:** Run after a new user is created and online. Some commands (like bfedit) can't be used until the user is online, so this is the place for those types of commands.
- **PostSetup.txt**: Run after a new user completes the Setup wizard.
- **Startup scripts:** Run each time RSIGuard starts up (i.e. to perform actions you want to occur each time a user starts using RSIGuard). There are 3 different scripts that can be run on startup (all of which by default are located in the installation folder).
    1. **PreInit.txt** – This script runs before RSIGuard itself begins. It can contain only very simple commands. For example, it might contain a command like "wait 5" to wait for a network resource to become available after login before RSIGuard starts.
    2. **Init.txt** – This script runs after RSIGuard has started and can be used to affect global settings.
    3. **Startup.txt** – This script runs after the RSIGuard user is online and can therefore be used to modify user settings such as break frequency. You can change the default location of this script with a command (e.g. in the registration script) like: "set m\RSIGuard\StartupFile S:\Net\RSIGuard\start.txt"
    4. **Online.txt –** This script runs each time a user comes online. If a new profile is created, or a user switches from one to another, this script is executed.
- **Break scripts:** A script can be configured to run before a break starts, and when a break is completed (i.e., not skipped or postponed). To configure these scripts to run, you must use the commands:
  "set m\RSIGuard\BeforeBreakScript BEFORE.TXT" and "set m\RSIGuard\AfterBreakScript AFTER.TXT"
  where BEFORE.TXT and AFTER.TXT should be replaced with the path to the scripts you want run.

Line Preprocessing Commands

Before each command line is executed, RSIScript preprocesses the line by replacing various escape sequences with processed text. This allows for string manipulation, logic control, calculation, etc. The following escape sequences are recognized.

date usage

- $d the following integer value should be replaced by the date/time it represents
- $Date() returns the current datetime stamp. $date(DAYS) returns the stamp of today + DAYS (which can be negative)
- $DateStr() returns a date in a specified format. With no parameters, it returns the current date and time in a default format (`%A %B %d %Y %H:%M:%S`). $datestr(FORMAT) returns the current date and/or time in a format you specify using a standard method of describing date-time formats described at http://www.rsiguard.com/help/help-datestr.htm. $datestr(FORMAT,TIMESTAMP) returns the date represented by TIMESTAMP formatted using the specified FORMAT string. You can use $Date() to generate a TIMESTAMP value.
- $duration(NUMSECONDS) – turns a number of seconds into a reasonable presentation

variable usage

- $1-$9 are replaced by the values of the 9 general usage variables assigned via the setvar command (i.e. aliasing)
- $val(SETTING,[DEFAULT-VALUE]) – substitutes with the specified RSIGuard setting (see *set* command for details on specifying SETTING). If SETTING does not exist, DEFAULT-VALUE will be used if present.
- $sSETTING is replaced by value of SETTING (e.g. $sm\RSIGuard\DataFolder). This escape sequence is obsolete and remains for backaward compatibility – use $val() instead.
- *$env(ENVVAR) – substitutes with the specified environment variable (see *setvar* command)

math & logic operations

- $plus(OP1,…OPN), $minus(OP1,…OPN) –returns (op1+op2…+opN) or (op1-op2-…opN)
- $times(OP1,OP2), $divide(OP1,OP2) – returns (op1Xop2X…opN) or (op1/op2/…opN)
- $rand() – returns a random number, $rand(N) – random # from 0 to N, $rand(A,B) – random # from A to B
- $min(OP1,…OPn), $max(OP1,…Opn) – returns the minimum or maximum of the listed operands
- $bound(OPmin,OPval,OPmax) – returns OPval, bounded to the range [OPmin,OPmax]
- $between(LOW,VALUE,HIGH) – returns 1 if LOW<=VALUE<=HIGH, else 0
- $round(Value,Precision) – returns Value rounded to no more than Precision digits (e.g. $round(3.178,1)=3.2)
- $equal(OP1,OP2) – substitutes a “1” if OP1 equals OP2 or a “0” if not (i.e. 1=TRUE, 0=FALSE).
- $lessthan(OP1,OP2), $lessthanequal(OP1,OP2), $greaterthan(OP1,OP2), $greaterthanequal(OP1,OP2) – substitutes a “1” if OP1 is “less than”, “less than or equal to”, “greater than”, or “greater than or equal to” OP2 – or “0” if not.
- $or(OP1,…OPN) – substitutes a “1” if any of OP1-OP-N are not “0” (i.e. if any OP is true).
- $xor(OP1,OP2) – substitutes a “1” if either OP1 or OP2 is not “0”, but not both (i.e. if one OP is exclusively true).
- $and(OP1,…OPN) – substitutes a “1” if OP1-OP-N are all not “0” (i.e. if all OPs are true).
- $not(OP1) – substitutes a “1” if OP1 is “0”, otherwise, substitutes a “0” (i.e. toggles OP between true and false).
- $if(CONDITION,TRUESUB,FALSESUB) – substitutes FALSESUB if CONDITION is “0”, otherwise TRUESUB.
- $switch(CASE,CASE-1:VALUE-1,…,CASE-N:VALUE-N,default:DEFAULT-VAL) – substitutes VALUE-X when CASE=CASE-X. If no CASE-X value equals CASE, substitutes with DEFAULT-VAL (or blank if no default).

string operations

- $GetArg(STR,N,[SEPARATORS]) – return Nth word in STR. Specify SEPERATORS to replace typical separators.
- $Left(STR,NUM)/ $right(STR,NUM) – replaces with first/last NUM chars of STR
- $Len(STR) – replaces with length of STR in characters
- $Mid(STR,NUM1,[NUM2]) – replaces with substring starting at NUM1’th position, using NUM2 chars (or going to end of string). NUM1 is 0-based (i.e. $mid(STR,0) is entire string).
- $IsDigits(STR) – returns 1 if all characters in STR are 0-9, otherwise 0
- $GetAt(STR,NUM) – returns the character at position NUM (0-based) in STR
- $Contains(HAYSTACK,NEEDLE) – return 1 if string NEEDLE appears in HAYSTACK. Otherwise 0.
- $Extract(STR,SEPARATOR,INDEX) – return the INDEXth (0-based) item in STR separated by SEPARATOR. E.g., $Extract(192.168.0.5,".",0) is 192. $Extract("Smith,John,jsmith",",",2) is jsmith.
- $Translate(STR) – returns translation of English string STR to current language if the translation is available
- $JSON(FIELD,[CONTENT]) – parse a JSON structure. If CONTENT is omitted, it uses the previous CONTENT. If the field is an array, you can use a FIELD like MyItem[4]. If FIELD is an object, result is the object as JSON.

special characters

- $$ is replaced by the $ symbol itself
- $ch(NUM) – substitutes the character of the specified character code (e.g. $ch(65)=A)

system information

- $sys(PARAMETERS) – Provides various information about a computer system as described below (Windows only):
  - Keyboard Information:
    - $sys(MoveableKbd) – return 1/0 if a moveable keyboard is present (e.g. something other than the built-in keyboard of a notebook computer).
    - $sys(KbdDevID,index) – returns vendor/product ID for a keyboard (index is 0-based).
    - $sys(NumKbds) – returns number of keyboards (count includes built in notebook keyboard).

  - Pointer-device Information:
    - $sys(NonUSBPtrIdx) – returns the index of a built-in pointing device (e.g. a notebook's touchpad or a desktop's PS/2 mouse) or -1 if no non-USB pointer is present. If multiple non-USB pointers are installed, this will only identify one of them.
    - $sys(PluggedInPtr) – returns 1/0 if a pointing device is plugged in through a USB, PS/2, serial, or bus port (i.e. a moveable pointing device).
    - $sys(MoveablePtr) – return $sys(PluggedInPtr) for notebook computers, or 1 for desktop computers.
    - $sys(PtrDevID,index) – returns vendor/product ID for an installed pointing device (index is 0-based).
    - $sys(PtrManuf,index) – returns manufacturer name (if available) for an installed pointing device (index is 0-based).
    - $sys(PtrInterface,index) – returns how an installed pointing device is connected to computer (index is 0-based). 1=Other, 2=Unknown, 3=Serial, 4=PS/2, 5=Infrared, 6=HP-HIL, 7=Bus mouse, 8=ADB, 160=Bus mouse DB9, 161=Bus mouse micro DIN, 162=USB.
    - $sys(PtrType,index) – returns pointer type for an installed pointing device (index is 0-based). 1=Other, 2=Unknown, 3=Mouse, 4=Track ball, 5=Track point, 6=Glide point, 7=Touch pad, 8=Touch Screen, 9=Mouse (optical sensor).
    - $sys(NumPtrs) – returns number of installed pointing devices.
    - $sys(CursorPos,[index]) – returns the coordinates (x y) where the mouse is pointing. You can move the mouse somewhere and restore its position (e.g. setvar 1 $sys(CursorPos) ; *some commands to move the mouse* ; mouse absmove screen $1). If index is 0 or not specified, the return is in form "x y". If 1, "x". If 2, "y". If 3, "x,y". So, for example, $between(100,$sys(CursorPos,1)),200) returns 1 (true) if the mouse pointer's x coordinate is in the range of 100 to 200.
    - $sys(LeftHandPtr) – returns TRUE if Windows has reversed the left/right logic of the mouse buttons.

  - Computer Information:
    - $sys(CompForm,index) – returns 0, 1 or 2 for "Unknown format", "Desktop" or "Notebook" respectively. Generally index can only be 0.
    - $sys(CompType,index) – returns 1-24 for various formats (e.g. 1=Other, 2=Unknown, 3=Desktop, 4=Low-profile desktop, 7=Tower, 8=Portable, 9=Laptop, 10=Notebook, 12=Docking station, etc.)
    - $sys(CompTypeName,index) – returns the text name associated with $sys(CompType)
    - $sys(CompManuf,index) – returns the manufacturer of the computer as a string.
    - $sys(CompWeight,index) – returns the weight of the system enclosure.
    - $sys(NumComps) – returns number of computers in system (would normally always be 1).
    - $sys(Docked) – returns 1 if a notebook is docked, or 0 if not (desktops always return 1).
    - $sys(TimeZone) – the timezone in which this computer is physically located

- Monitor/Screen Information:
  - $sys(NumMonitors) – returns the number of monitors comprising the desktop (e.g. a system with 2 monitors that display the same thing would report 1 monitor, or with 2 connected screens but only 1 selected to be part of the desktop would report 1).
  - $sys(NumScreens) – returns the number of screens attached. 2 monitors displaying the same thing, or a system with 2 monitors with only one in use by the desktop would still report 2 screens.
  - $sys(NumVidControllers) – returns the number of video controllers in the computer, whether or not there are actually monitors hooked up to these controllers.
  - $sys(ScreenName,index) – returns name of specified screen.
  - $sys(ScreenOrientation,index) – returns orientation of screen -----.
  - $sys(ScreenFrequency,index) – returns refresh frequency of specified screen.
  - $sys(ScreenPrimary,index) – returns 1/0 if specified screen is/isn't primary monitor.
  - $sys(ScreenActive,index) – returns 1/0 if specified screen is/isn't active and enabled.
  - $sys(ScreenID,index) – returns device name ID of specified screen.
  - $sys(ScreenWidth,index), $sys(ScreenHeight,index) – returns width/height (in cm) of the specified screen (works even if screen is not part of desktop).
  - $sys(ScreenHRes,index), $sys(ScreenVRes,index) – returns current, software-selected resolution in pixels of screen (returns 0 if screen not part of desktop).
  - $sys(ScreenOriginX,index), $sys(ScreenOriginY,index) – returns current, top-left origin in coordinate space of screen (returns 0 if screen not part of desktop).
  - $sys(TotalDispWidth) – returns the sum of the widths (in cm) of 1 or more attached monitors.
  - $sys(MaxMonHeight),$sys(MinMonHeight) – returns the tallest/shortest monitor height (in cm).
  - $sys(IconFontHeight,index) – returns estimate of height of icon font in mm on screen (baseline to baseline).
  - $sys(FontWidth,fontindex), $sys(FontHeight,fontindex) – returns size of an M in specified system font in pixels (0=icon font, 1=message font, 2=menu font)
  - $sys(FontName,fontindex) – returns name of system font (0=icon font, 1=message font, 2=menu font)

- Network/User Information:
  - $sys(Network) – gets name of current network (e.g. LAN Connection)
  - $sys(Connected) – returns 1/0 if you are connected to internet, tested by accessing www.google.com
  - $c is replaced by 0 or 1 depending on whether or not the computer is connected to the internet
  - $sys(IPAddress) – returns IP address of your computer
  - $sys(WirelessProfile) – returns name of currently connected wireless network (if connected)
  - $netinfo(API,VALUE) – returns result from calls to GetComputerName() (API=1) and GetComputerObjectName() (API=2) (see ShowCompInfo.exe at http://www.rsiguard.com/support/it_staff.htm). Contact RSIGuard support for more information on how this can be used to determine domain, computer name, and other information.
  - $sys(OESPing) – contacts OES and returns "OK" if successful, or an error message otherwise.
  - $sys(UserEmail) – returns an email address for the current user if available
  - $sys(AdminEmail) – returns an RSIGuard administrator's email if available
  - $sys(IPConfig,VALUE) – returns information about adapters. The values 0 to 7 return information about the first active WiFi adapter (0:Adapter Name, 1:Adapter Description, 2:Adapter Address, 3:IP Address, 4:IP Mask, 5:Gateway, 6:DHCP Enabled, 7:DHCP Server). INDEX+16 returns the same values for the first active Ethernet adapter.

- RSIGuard/System Information:
  - $sys(HTMLWin) – tells if the RSIGuard HTML window is being displayed.
  - $sys(ExeFolder) – returns the folder where the RSIGuard executable is located.
  - $sys(AppPath) – returns the full path to the RSIGuard (or Guardian) executable.
  - $sys(ReleaseDate) – return the date that the current version of RSIGuard was released.
  - $sys(Sys32Folder) – (Windows) returns the system32 folder
  - $sys(ResourcesFolder) – returns the location of the resources folder
  - $sys(DataFolder) – returns the location of the usage data folder
  - $sys(ScriptsFolder) – returns the location of the scripts folder
  - $sys(OneDriveFolder,[INDEX]) – returns folder of OneDrive mirror folder. Index=-2 for Personal OneDrive, -1 for Business OneDrive, 0 for alternate default OneDrive folder, or N for Nth account.
  - $sys(NoticeState) – returns integer representing Window's concept of whether or not it's appropriate to interrupt user based on http://msdn.microsoft.com/en-us/library/bb762533%28v=vs.85%29.aspx
  - $sys(NoticeOK) – returns BLANK if user is ready for a notice, or a string explaining why they aren't.
  - $sys(RemoteSession) – returns 1/0 if RSIGuard is running in a remote session
  - $sys(ForegroundApp) – returns the titlebar of the foreground application.
  - $sys(ForegroundClass) – returns Windows class (Windows only) of the foreground application.
  - $sys(FullScreenFgrnWnd) – returns if the foreground window is in full-screen mode.
  - $sys(DisplayForcedOn) – returns 1/0 if computer is being prevented from turning off (e.g., 1 if the user is watching a video or in a presentation)
  - $sys(SystemMetric,INDEX) – returns system metric based on INDEX from http://msdn.microsoft.com/en-us/library/windows/desktop/ms724385(v=vs.85).aspx

$sys(Refresh) –insures data is current (normally it may be up to 10 seconds old) (returns nothing).

user information

- $i is replaced by the users RSIGuard ID#
- $m is replaced by the users unique OES or RSIGuard GUID
- $n is replaced by the name of the current user profile (e.g. Joe Smith)
- $p is replaced by the profile subkey string for the current user (see the 'set' command below) (e.g. _JOESMITH)
- $u is replaced by the username of the currently logged in Windows user (e.g. jsmith)
- $UserGroup(0/1) – returns Windows user group (Windows only). If 0, returns local groups. If 1, global groups.
- $UData(WorkSecs/MouseSecs/KbdSecs/AvgTimeBetweenBreaks/AvgBreakLength/PoliteMode/WillpowerLevel/WorkRestriction/ScheduledBreaks/WRSecsRemaining/IdleSecs/SecsSinceStartOfWork/SecsTilNextBreak) – returns current values of DataLogger and Settings data for the specified field.
- $Data(FIELDID,NUMDAYS,TOTALS) – returns DataLogger data (like $udata()) for a much wider selection of DataLogger fields. If NUMDAYS is omitted (or 0) then the current value of the field is returned. Otherwise, the returned value is for the preceding NUMDAYS (not including today). If TOTALS is omitted or 0, then the returned value is a weighted average based on a 5-day work schedule. If TOTALS is 1, then the returned value is the sum over the preceding NUMDAYS. The FIELDID is a 2-character ID. E.g. CD=Overall Break Compliance, CI=Microbreak Compliance, TH=Hours of activity. For other values, see Appendix 2 of www.rsiguard.com/documents/help/DataLoggerAnalysis.pdf.

version checking

- $r is replaced by the RSIGuard Edition/Version (e.g. RSIGuard Stretch Edition v3.0.37)
- $version(v/n/p/s/f/e/1/2/3/4/b) – return RSIGuard version. Sample output for v=v5.0.16c, n=5.0.16c, p=v5.0.16cSB, s=Stretch Edition v5.0.16cSB, f=RSIGuard Stretch Edition v5.0.16cSB, e=Stretch Edition, 1/2/3/4=version parts, b=build version (e.g., v6.1.6.2.20220801).
- $v for MFC versions, returns the Windows version (95=4.0 98=4.10 ME=4.90 NT3.51=3.51 NT4.0=4.0 2000=5.0 XP=5.1, Vista=6.0, Windows 7=6.1, Windows 8=6.2, see http://msdn.microsoft.com/en-us/library/ms724832%28v=VS.85%29.aspx for complete list). For CPE versions, returns Mac OSX version followed by 'M' (e.g. 10.49M) or Windows version followed by 'W', or Linux version followed by 'G'.
- $w for MFC versions, returns 0 if Windows platform is Win95/98/ME or 1 if Windows platform is WinNT/2000/XP. For CPE versions, returns '1' for Windows or 'M' for Mac OSX or 'L' for Linux.
- $RedHat() – returns the RHEL release (e.g. 6.1) (only works for RHEL installations)
- $Framework() – returns 'M' for MFC, 'C' for CP versions. (v4.0.26i and later)
- $EmbeddedHTML() – returns true if the 'html open' command uses an embedded browser that supports <a> tags with RSIScript allowed in the href value. Generally, this is supported on all recent versions of RSIGuard for Windows.

other commands

- *$q – reports the method RSIGuard is using to record computer-wide settings
- *$Readable(FILEPATH) – returns 1 if the FILEPATH is readable, 0 if not (or doesn't exist). This can also be used to test if a file exists. E.g., $if($readable(FILEPATH),read FILEPATH)
- *$Writeable(FILEPATH,TESTTIME) – returns 1 if FILEPATH is writeable, 0 if not. TESTTIME (in seconds) must be >=1 – this routine will keep testing the FILEPATH for at least that many seconds.
- $ExeDir() – returns folder where RSIGuard executable is located (same as $sys(ExeFolder))
- $cwd() – returns folder current script is in. Can be used to ensure absolute file paths to files in same folder, e.g. for read commands, $InFile(), $FindImg(), etc. For example, to read file abc.txt in the same folder, do "read $cwd()/abc.txt"
- *$InFile(NEEDLE,FILE) – substitutes a "1" if the string NEEDLE appears in file FILE. Since v4.0.41, this test is case insensitive. Since 5.0.1, this test ignores excess spaces. NEEDLE can end in * to match any line that starts with NEEDLE.
- *$IsKey(KEY) – returns "1" if key is pressed. For example, $iskey(A) is 1 if the 'A' key is down. Special keys require codes. E.g. $iskey($ch(17)) tells if the Control key is pressed. $iskey($ch(13)) for Return, or $iskey($ch(27) for ESC.
- *$IsRunning(TITLEBARTEXT) – substitutes the number of windows open that contain the specified title bar text (see example 5 below).
- *$IsProcess([PROCNAME]) – returns 1 or 0 if specified process is found. An '*' can be placed at the start, end or both to match process names that end, start or contain the specified string. If PROCNAME is omitted, a list of currently running processes is displayed.
- *$IsClass([?]CLASSNAME,[WNDNAME]) – returns # of instances of class found running. If classname starts with ?, it displays a list of matching classes running. If [WNDNAME] is specified, returns # of instances of class found running associated with a window of specified name.
- $event(event/data/FIELD) – returns event name, data block, or field value from the last event (for use in 'event action')
- *$AppAtPoint([x/CX],[y/CY],[TITLE/CLASS/ITEMNAME/HANDLE]) – returns the requested information about what's at the specified screen coordinates (CX means x-coordinate of pointer, CY means y-coordinate of pointer). By default, TITLE is selected, which returns the name of the application at the point. CLASS returns the OS name of the class of window at the point. ITEMNAME returns the text associated with the object right at the point. HANDLE returns the OS ID for the object at the point. If no coordinates are specified, the mouse pointer (CX,CY) is used. This function is useful for AutoClick filters to control special behaviors based on what is being clicked on.
- *$FindObj(OBJECTTEXT) – looks for OBJECTTEXT in current window. If found, it returns the X,Y coordinates of the center of the text (e.g. for use with 'mouse absmove' command.
- *$FindImg(IMAGEFILE,curwnd/desktop,[TOLERANCE]) or $FindImg(IMAGEFILE,rect,LEFT,TOP,RIGHT,BOTTOM,[TOLERANCE]) or $FindImg(IMAGEFILE,window,WINDOWNAME,[TOLERANCE]) – scans for occurrence of image IMAGEFILE (must be .bmp format) in the specified region of the screen. "curwnd" scans the foreground window, "desktop" scans the entire desktop across all monitors, "rect" scans the specified rectangle, and "window" searches the window of an application whose title contains WINDOWNAME. If found, $findimg() returns the X,Y coordinates of the center of the image (e.g. for use with 'mouse absmove screen' command). Otherwise it returns 0. If TOLERANCE is not specified, search only matches if image appears exactly. If TOLERANCE is specified and greater than 0, pixels can be up to TOLERANCE distance apart and still "match". Distance is the linear distance of the RGB values in 3D space (i.e. $((R2-R1)^2+(G2-G1)^2+(B2-B1)^2)^{½}$. A reasonable value to address variation in different renderers might be 10, though experimentation may be necessary. Tolerance is available in v5.0.6f or later.
- *$FindWnd(WINDOWTITLEMATCH) – returns the first window ID that matches the specified string. WINDOWTITLEMATCH's format is the same as the format for WINDOWTITLE in the 'activate' command. See that command for complete documentation.
- *$BTFilter(FORMAT) – tells if a BreakTimer filter condition is true. If FORMAT is 1, the name of any active rule is shown. Otherwise, 0, 1, or 2, for no match, polite mode match, or BreakTimer disable match.
- $notice() – returns 1 if a notification is on screen, or 0 if not.

- $do(OP1,…,OPn) – Performs OP1…OPn as commands. You can also separate multiple commands with semicolons.
- $api(URL[,PREFIX[,ERRCMD,[POSTDATA]]]) – Returns the value returned by a GET/POST request to URL. $api(URL) returns the response to a GET request to URL. $api(URL,PREFIX) requires that the response to the GET request begin with PREFIX, and the value returned is content after the PREFIX. If response doesn't start with PREFIX, $api() returns #ERR#NoPrefix. $api(URL,PREFIX,ERRCMD) executes ERRCMD RSIScript if either a server error occurs or the PREFIX (if specified and not blank) is missing. $api(URL,PREFIX,ERRCMD,POSTDATA) sends request as a POST with POSTDATA as the content.
- $choice(PROMPT,OPTION1,…,OPTION-n) – user is given option to press a single key to select between options, and $choice() is replaced by the selected option. E.g., type $choice(Which signature?,Sincerely,Thanks),<enter><tab>Joe
- $ClipboardText() – returns the contents of the text "copy buffer" (i.e. clipboard). If you select text and do a "type <ctrlc>" to copy it, then you can use $ClipboardText() to use that copied text.
- $Language() – returns RSIGuard's 3 character code for RSIGuard's current language
- $LanguageCode() – returns RSIGuard's 2 character code for RSIGuard's current language
- $LanguageName() – returns RSIGuard's name for RSIGuard's current language xzx
- $LanguageOSID(INDEX) – returns Windows locale (0), current language (1), current UI language (2).
- $Live() – replaced with default URL for live content links
- $URLEnc(URL), $urldec(URL) – encodes or decodes URL text (e.g. a space is encoded as %20).
- $g is replaced with a randomly generated GUID.
- $Registered() – replaced with 0 if RSIGuard is unregistered, 1 if registered w/out stretches, 2 if fully registered.
- $IsCloudConnected() – returns 1 if RSIGuard is connected to Cloud, 0 if not.
- *$View(CONTENT,EXTENSION) – view CONTENT with the default handler for specified EXTENSION.
- $recursive(0/1) – Normally, RSIScript processes $*command()* items, and then also processes the result. This can produce recursive processing which can be helpful. If you don't want this to happen, placing $recursive(0) within a command prevents further-to-the-right processing. So, appending $recursive(0) at the end prevents all recursive processing for the command.

Example1: ask a user with the high willpower setting if they'd like to switch to medium willpower
$if($equal($val(u\$p\BreakTimer\WillpowerLevel),0),"$do(msg text $u you might want to switch to Medium willpower,msg send)")
This uses $equal() to test if the current willpower setting is 0, which is the high-willpower setting. If so, it replaces the command line with the $do() statement. Note that quotes are needed around $do() since we don't want the $do() preprocessing step to occur unless the condition is true. Alternately, the $do statement could have called another script that uses the 'question' command to actually change the willpower setting.

Example 2: automatically register pre-selected users, and tell other users this is a trial
$if($infile($u,s:\RSIGuard\userlist.txt),"$do(register xacme1234)","$do(msg text "Tell your supervisor if you want to use RSIGuard",msg send)")
This uses $infile() to test if a user's login name appears in a text file (s:\RSIGuard\userlist.txt in this example). If the name appears in the file, the user is registered with the "register xacme1234" command. Otherwise, they are allowed to continue but told to tell their supervisor if they wish to use RSIGuard.

Example 3: automatically determine who should run RSIGuard from a username list
$if($not(infile($u,s:\RSIGuard\userlist.txt)),"$do(exit)")
This uses $infile() to test if a user's login name appears in a text file (s:\RSIGuard\userlist.txt in this example). If the name is not in the file, RSIGuard exits. This can be used to control who uses RSIGuard by scripting that all users run RSIGuard, but only those in the list will continue to run past the initial start up.

Example 4: disable ability to send Health Status Reports for people not in the Engineering department
$if($not($equal($val(u\$p\Reports\Department),Engineering)),"$do(set u\$p\GeneralSetup\AccessSendHSR 0)")
This uses $not($equal()) to test if user's Deparmtent is<u>n't</u> Engineering, & if not, disables ability to send Health Status Reports.

Example 5: if Microsoft Word is running, activate the window (i.e. bring to front), otherwise launch Microsoft Word
$if($IsRunning(Microsoft Word),activate wnd Microsoft Word,open winword.exe)

RSIScript Command Descriptions

***activate** – bring a specified window to the frontmost (activated) position

Syntax options:

activate wnd/firstwnd/showlist [WINDOWTITLE] [WINDOWCLASSNAME]
activate winid WINDOWID
activate WINDOWTITLE *(this form is deprecated and should no longer be used)*

Notes:

This command allows you to bring a window/application to the foreground (e.g., to ensure a 'type' command puts text in the right window). The WINDOWTITLE is compared to the name that appears in the Window Title bar. There are 3 modes of comparison. If WINDOWTITLE starts with:

- ! – the match must be exact, including case. E.g., if the Window shows **Notepad - test.txt** then it would only match if WINDOWTITLE were **!Notepad - test.txt** (**!Notepad** or **!Notepad - TEST.txt** would not match).
- $ – WINDOWTITLE is an RSIScript expression to match windows. In your expression, window title is inserted wherever $!$ appears, window class is inserted wherever C appears, and windows show state (Normal, Hidden, Minimized, Maximized) is inserted wherever S appears. Since you don't want the expression evaluated, it must be enclosed in {{ }}. And if the expression contains spaces, quotes are needed. E.g., you could activate a Notepad window with a file that starts with 't' using the command **activate firstwnd "{{$equal($left($!$,11),Notepad - t)}}"**. If there is an error in your RSIScript syntax, no windows will match. So if you seem to not get expected matches, check your RSIScript using showlist (see example below).
- any other character – if WINDOWTITLE appears in the title, the window matches. E.g., **activate wnd Notepad** would activate any window containing the word Notepad (not case-sensitive).

For example, to activate a Notepad Window with an untitled file, you could use "activate wnd Untitled - Notepad". If there are multiple windows that match the window title, a popup screen appears to let you select which of the windows you want to activate. If instead you use the 'firstwnd' option, the first window in the list will automatically be activated (to prevent the popup screen). If the 'showlist' option is specified, a list will be shown (in Notepad or Excel) with the open windows, along with their class names, to help you craft a more precise 'activate wnd/firstwnd' command for the window you need activated.

Windows have internal "class names" (shown by the showlist command) that can be used to further narrow the specification of which window to activate.

Note, since this command has up to 4 arguments, the 3rd argument (WINDOWTITLE) must be enclosed in quotes if it contains spaces.

Alternately, activate can use the results of a $findwnd() operation. E.g. **activate winid $findimg(Notepad)** is the same as **activate firstwnd Notepad**. The advantage is that this allows different behavior if the window is found or not. E.g., **setvar 1 $findimg(Notepad);$if($1,activate winid $1;type Hello)** would activate Notepad if present and type Hello into it. But if there is no instance of Notepad running, no typing would occur.

Examples:

activate wnd {{$and($contains($!$,Notepad),$equal($S$,Minimized))}} # activate a minimized Notepad window
activate wnd {{$and($contains($!$,Notepad),$equal($C$,Notepad))}} # activate Notepad window, filters out other windows that might have the word Notepad in their title
activate firstwnd "{{$equal($left($!$,11),Notepad - t)}}" #activate the first Notepad window found with an open file that starts with a 't'
activate showlist {{$contains($!$,Notepad)}} # shows a list of all windows containing the word Notepad
activate showlist {{$contains($!$)}} # shows list of resulting windows. Since there's an error in the RSIScript, it shows: Error 'Incorrect number of operands (should be 2)' was reported in the RSIScript expression: $contains($!$)
setvar 1 $findwnd({{$equal($Right($!$,7),Notepad)}});$if($1,activate winid $1;type Hello) #type Hello into Notepad

***addkckey** – create hotkeys

Syntax options:

addkckey KEY ID CATEGORY OPTIONS NAME

Notes:

This command allows you to create hotkeys. Using this command requires assistance from RSIGuard support. KEY is a binary mask for Ctrl/Alt/Shift/Win followed by the key's keycode (e.g. 101065 for Ctrl-Shift-A). A list of keycodes is shown in http://www.rsiguard.com/documents/help/DataLoggerAnalysis.pdf in the "Extended DataLogger Tools" section. ID is an internal ID (256=Single Click, 257=Double Click, 258=Triple Click, 259=Right Click, 260=Left DragLock, 270=Right DragLock, 271=Skip Next Click, 272=Middle Click, 273=Middle Drag Lock, 512=RSIGuard-To-Front, 513=Hide RSIGuard, 514=Take A Break Now, 515=Show ForgetMeNot Now, 516=Enable/Disable AutoClick, 32768 for any other hotkey). CATEGORY is the hotkey category (0=RSIGuard Function, 1=Launch Application, 2=Open File, 3=Open Web Page, 4=Type Text, 5=Insert Text File, 6=Run RSIScript File, 7=Do RSIScript Command, 8=Windows Operations). OPTIONS is the "argument" to the hotkey (use "" if no argument is needed). Name is the hotkey's name.

Example:

add hotkey to run a script. KEY=1100070=Ctrl+Alt+F, ID=32768 (use this same ID for any non RSIGuard function)
CATEGORY=6=Run RSIScript File, OPTIONS=*the-file-to-run* NAME=*the descriptive name you want for the hotkey*
addkckey 1100070 32768 6 "S:\RSIGuardNetwork Scripts\Example.txt" "Run Example Script"

---

addprofile – adds a new user profile

Syntax options:

addprofile PROFILENAME

Notes:

Adds a profile. Generally, this command should only be used in an automated user-setup process. It is not a recommended way to create new profiles. You can switch between profiles using *setprofile*

****admin** – enable/disable admin settings/menu access

Syntax options:

Admin 1/0

Notes:

This command is only accessible to scripts that originate from the OES.

---

alertset – change RSIGuard settings and notify user of change

Syntax options:

alertset SETTING number/text VALUE (set the value of the RSIGuard setting specified by SETTING to VALUE)

Notes:

This command is identical to the 'set' command described below, except that, if the current value of SETTING is not already VALUE (i.e. the setting needs to be updated), the user will receive a message letting them know that the setting is being changed.

See also: fset, queryset, set

---

beep – make a standard system sound

Syntax options:

beep

---

bfedit – modify the list of BreakTimer filters

Syntax options:

bfedit add active/running disable/polite APPLICATION-TITLEBAR-TEXT
bfedit delete APPLICATION-TITLEBAR-TEXT
bfedit deleteall

Notes:

Commands to add and remove BreakTimer filters.
To add a filter, use a command like “bfedit add running disable PowerPoint Slide Show”. The ‘active’ option means this filter applies when the application is the active foreground application. The ‘running’ option means this filter applies if the application is running at all. The ‘disable’ option means that while the condition (active or running) is true, the BreakTimer should be disabled. The ‘polite’ option means that while the condition is true, BreakTimer should be in polite mode (where a “Break Needed” button must be clicked to start the break). The application name corresponds to the text in the title-bar of the application window.
To delete a filter, you need only include a fragment of the application name for the filter you wish to delete. For example, you could use either “bfedit delete PowerPoint Slide Show” or “bfedit delete PowerP”.
To delete the all BreakTimer filters, use “bfedit deleteall”.

---

break – initiate a BreakTimer break

Syntax options:

Break reason REASON-TEXT – show a BreakTimer break with REASON-TEXT shown as the reason for the break

---

,*changelanguage** – specify RSIGuard’s display language

Syntax options:

changelanguage 3-CHAR-LANG-CODE – set RSIGuard’s display language to the specified language

Notes:

Changes RSIGuard language and will normally restart RSIGuard automatically in the new language.

---

cli – change special characters in RSIScript

Syntax options:

cli showchars – show the current special RSIScript characters
cli setchars ABCDEF
cli setchars %AA%BB%CC%DD%EE%FF
cli pop – restore the default characters

Notes:

By default, RSIScript has 6 default special characters used in processing: " ( ) ; , $
This advanced command lets you replace those characters with the specified 6 characters. You can specify the characters directly, or as a url-encoded string. You should only use this command if you really know what you’re doing!

Examples:

cli setchars |[];,^
cli setchars %24%25%20%22%28%27

***cloud** – interact with RSIGuard cloud

Syntax options:

cloud <various options>

Notes:

Used to connect to, and interact with the RSIGuard Cloud system.

***crash** – crash RSIGuard

Syntax options:

crash

Notes:

Used to test the crash information reporting system.

***datafile** – perform operations related to DataLogger data files

Syntax options:

datafile merge SRCFILE1 SRCFILE2 DESTFILE

Notes:

Takes the data in SRCFILE1 and merges it into SRCFILE2, with the result going into DESTFILE.

***debug** – enables logging of debug information about scripts being run

Syntax options:

debug 1/0

Notes:

If 1, log is written to %temp%/RSIScriptDebut.txt. Be sure to disable when not needed or debug data will be written for all subsequent scripts. Defaults to '0' when RSIGuard is run.

---

discomfort – notate where a user is experiencing discomfort for HSRs

Syntax options:

discomfort WristHand/NeckUpperBack/Shoulder/ElbowForearm/WristHand 1/0

Notes:

Tells RSIGuard to set discomfort points associated with the specified area to be set (1) or unset (0). This command is for use by third-party applications that wish to set HSR values.

---

doonce – do a command once

Syntax options:

doonce Profile/User/Machine ID CMD

Notes:

Does RSIScript command CMD once within the context of Profile:RSIGuard-user-profile, User:user-login, or Machine. For example, "doonce User test001 msg text hello;msg send" will display "hello" the first time is is executed only.

***ergocoach** – internal command to operate ErgoCoach functions

Notes:

Used internally by ErgoCoach features.

,*event** – edit RSIGuard events

Syntax options:

event action/clear EVENTNAME RSISCRIPT-ACTION
event fire EVENTNAME DATA

Notes:

Creates (event action) or clears (event clear) actions associated with RSIGuard events. ‘event fire’ triggers an event. DATA is in name-value-pair format (e.g. val1=5&val2=xyz).

Examples:

notify the OES when new connections to wireless networks are created
event add WiFiProfChange oes fwdevent

---

exit – terminate RSIGuard

Syntax options:

exit

Notes:

Exits RSIGuard.

See also: return

.*factoryreset** – reset all RSIGuard settings to initial install state

Syntax options:

factoryreset

Notes:

This command deletes all settings and so should only be used if you wish to remove all previous settings changes.

---

fmnedit – modify the list of ForgetMeNots

Syntax options:

fmnedit add FORGETMENOTMESSAGE
fmnedit delete FORGETMENOTMESSAGEFRAGMENT
fmnedit deleteall
fmnedit addex DISPLAYFREQUENCY SOURCECHARACTER FORGETMENOTMESSAGE
fmnedit deletefromsource SOURCECHARACTER

Notes:

Commands to add/remove ForgetMeNot messages. To add a message, use a command like “fmnedit add Remember to rest”. You can add line-breaks by embedding
 within the message (e.g. ‘fmnedit add Remember
***This
***That’). This also cause messages to be drawn left-justified instead of centered.
To delete a message, you need only include a fragment of the message you wish to delete. E.g., use either “fmnedit delete Remember to rest” or “fmnedit delete Remember” or “fmnedit delete to rest” to delete the message added in the example above. To delete the entire list of ForgetMeNots, use “fmnedit deleteall”.
‘fmnedit addex’ allows you to add a ForgetMeNot and specify the frequency of display each rotation (0-99, 0 means 0% chance of display, 99 means 99% - 30 is default value for ‘fmnedit add’). It also lets you specify a source/group for the item. The source/group can be ‘a’-‘z’ or ‘A’-‘R’ (‘S’-‘Z’ are reserve by RSIGuard). This lets you use ‘fmnedit deletefromsource’ to delete all items in the group. So you can add N messages in a group, then delete them all when done, in order to run “ForgetMeNot message campaigns”. Lowercase sources let the user enable/disable the messages. Uppercase sources are locked on. (Lowercase sources available in v6.0.2 or later only).

---

fmnmsg – pop up a ForgetMeNot style message

Syntax options:

fmnmsg MESSAGE

---

fset – remotely control RSIGuard settings and specify data type

Syntax options:

fset SETTING number/text VALUE (set the value of the RSIGuard setting specified by SETTING to VALUE)

Notes:

Exactly like the 'set' command except the number/text argument lets you specify if the data should be stored as a number or a string.

See also: alertset, queryset, set

---

html – open an RSIGuard-based html browser

Syntax options:

html open/dlg CAPTION 0/BUTTONTEXT WIDTH HEIGHT URL PREFIX – open a browser to PREFIX/URL
html navigate URL – if a window is open, this browses to PREFIX/URL
html close – close the browser window if open
html caption – changes the caption in the open browser
html resize WIDTH HEIGHT – change the size of the browser window

Notes:

Allows extended html browsing within RSIGuard. The 'html open' command CAPTION specifies the browser window caption. 'dlg' removes the ability to cancel the window (so a close button, or link to close the window must be provided). 0/BUTTONTEXT means either place a close-browser button at the bottom of the window with the specified text, or if 0, don't show a close button. WIDTH & HEIGHT specify the window dimensions in pixels (-1 selects a default value based on screen size). PREFIX tells the left hand side of the URL (e.g. www.rsiguard.com) and the URL is the place within that prefix (e.g. index.htm). They are specified separately to prevent (if desired) browsing away from a specified domain. If $embeddedHTML() returns 1, then within the html that is being browsed, you can create links of type "rsiguard:RSIScript". For example, if you have a link such as Say Hello, then clicking on that link would display an RSIGuard message that says 'hello'. If the width or height is -1 in an "html resize" command, that dimension won't be resized.

See also: open, $EmbeddedHTML(), $sys(HTMLWin)

***insertfile** – insert the contents of the specified text file into the current application

Syntax options:

insertfile FILENAME

Notes:

Inserts the text stored in the file FILENAME into the application with edit focus.

Example:

insertfile C:\My Documents\StandardEmail.txt

***integrate** – command to integrate HR data from a database into RSIGuard reporting

Syntax options:

Integrate enc/clr DATABASE

Notes:

The 'integrate' command causes RSIGuard to search the database file for HR data for the logged in user, and to store the relevant data. The 'enc' option states that the database has been encoded. The 'clr' option states that the database is clear text. The format for the database is described in the section 1.9 of the RSIGuard Program Administrator's Guide.

***log** – control the script log file

Syntax options:

log file FILENAME (specify a file to log script information to)
log error/info/msgs 1/0 (specify if errors/informational-messages/'msg log'-commands should be logged, default=1)

Notes:

The log file gets 3 kinds of messages. Error messages are added to the log file when a command in the script file is invalid or results in an error. Info messages are added by certain commands (e.g., the 'register' command logs when a user is first registered, and the 'set' command logs your changes to a user's setup). You can create your own messages and write them to the log file (or display them on the user's screen) with the 'msg' command.

Example:

```
# define where to write the log file
log file \\NETDRIVE\RSIGuard\log.txt
# this next line specifies that error messages should be discarded and not written into the log file
log error 0
```

---

oes – make requests from the OES system

Syntax options:

oes api POSTDATA
oes fwdevent

Notes:

This command is only available for use by the OES system.

---

mail – open a prefilled email

Syntax options:

mail cc/bcc EMAILADDRESS
mail subject SUBJECT
mail msg BODYTEXT
mail sendto [EMAILADDRESS] #if omitted, lets user type in address

Notes:

This command works with Outlook, but is not guaranteed to work with all mail clients. Specify the desired fields, and finish with 'mail sendto' to open the email.

Example:

```
#send a CC to someone
mail cc mymanager@acme.com
#ask which of 3 subjects to use
mail subject $choice(Subject,Your Request,Followup,Please Respond)
#we'll specify the recipient later
mail sendto
#wait a second for email to open
wait 1
#to address is in paste buffer, so paste it in after email opens
type <ctrlv>
```

---

*,**menuedit** – modify the list of Soft Menu Items

Syntax options:

menuedit add SOURCECHARACTER MENUNUM INSERTPOSITION “MENUITEMTEXT” COMMAND
menuedit deleteall
menuedit deletefromsource SOURCECHARACTER [MENUTEXT]

Notes:

Commands to add and remove soft menu items. To add an item, use “menuedit add”. The SOURCECHARACTER identifies the category of the menuitem for the purpose of deleting all items from a particular category (@ is reserved for custom RSIGuard installations). MENUNUM is 0 for Tools, 1 for Setup, 2 for Help. INSERTPOSITION tells where in the menu to put the item (1 is first, 2 is second, etc., and 0 means last item in menu). “menu deletefromsource” deletes all menu items in the specified category, unless MENUTEXT is specified, in which case only items whose text exactly equals MENUTEXT are deleted. “menu deleteall” deletes all soft menu items. Identical ‘menuedit add’ commands don’t add duplicate entries, but any change (position, name, or command) will create duplicates – so using the SOURCECHARACTER as a menu ID for deletefromsource commands can be used to prevent duplicates.

Example:

menuedit add G 0 0 “Visit RSIGuard Website” “open http://www.rsiguard.com”
menuedit deletefromsource G (deletes only the above menu item and others created with a source character of ‘G’

---

mode – change an RSIGuard mode

Syntax options:

mode ui rsiguard/oesdesktop/quiet/silent
mode edition stretch/callcenter

Notes:

The “ui” option specifies RSIGuard’s UI presentation. “RSIGuard” is the full user interface. “OESDesktop” shows the limited UI without BreakTimer, ForgetMeNots, AutoClick & KeyControl. “Quiet” also hides the UI in the system tray. “Silent” hides the UI entirely.

The “edition” option specifies whether or not RSIGuard is in regular “stretch edition” mode or “call center” mode.

***mouse** – control mouse movement and clicking

Syntax options:

mouse click/dblclick/rtclick/ldown/lup/rdown/rup/mdown/mup
mouse relmove delx dely
mouse absmove client/screen xpos ypos
mouse lefthanded 1/0/toggle

Notes:

The first usage line does mouse clicking. ‘click’ is a single left click. ‘dblclick’ is a double left click. ‘rtclick’ is a single right click. ‘ldown’ presses and holds down the left mouse button. ‘lup’ releases the left mouse button after an ‘ldown’. ‘mdown/mup’ and ‘rdown/rup’ do the same for the middle and right buttons. ‘mouse relmove’ moves the mouse the distance specified from its current position. Distance is specified in screen pixels. ‘mouse absmove’ moves the mouse to an absolute position on the screen. If the third argument is ‘client’ then the coordinates are relative to the client area in the currently active window. If the third argument is ‘screen’ then the coordinates are relative to the upper left corner of the screen. ‘mouse lefthanded’ lets you change both Windows & RSIGuard’s setting for left-handed operation to 1 (for left-handed), 0 (for right-handed), or toggle (to change between the two).

Example:

mouse absmove client 100 200 – move cursor 100 pixels to right and 200 down from top left of current window
mouse absmove screen $findobj(Browse) – move cursor to the Browse button in current window
mouse absmove screen $findobj(c:\documents\printicon.bmp) – move cursor to print icon in current window

mouse click – click mouse (e.g. after doing a mouse absmove command)

example to click on an icon, and if it’s found, click on it (this could be assigned to a KeyControl DoCommand hotkey)
setvar 1 $findimg(c:\tmp\tmp.bmp);$if($1,mouse absmove screen $1;mouse click)

---

msg – create messages that can be stored in the log file or displayed on a user's screen

Syntax options:

msg text TEXT (append TEXT to the current message buffer)
msg send (display the current message buffer on the user's screen and then clear the message buffer (i.e. 'msg clear'))
*msg log (write the current message buffer to the log file (but don't reset the message buffer (see 'msg logclr/clear'))
*msg log (write the current message buffer to the log file and then clear the message buffer)
msg clear (clear the message buffer. This only needs to be used between messages that are going only to the log file.)

Notes:

The 'msg' command allows you to format a message into a 'message buffer' for writing to the log file and/or displaying on a user's screen at startup time. You create a message by appending pieces of the message to the message buffer and then writing it to the log file ('msg log') or sending it to the user's screen ('msg send'). Sending the message to the user's screen with 'msg send' also clears the message buffer (in preparation for creating your next message). (The only reason you need to explicitly write 'msg clear' between commands that you write to the log file is that if you want to send a message to both the log file and the user's screen, you build the message buffer, then call 'msg log' then 'msg clear'. If 'msg log' also cleared the message buffer, you'd have to rebuild the message twice to send it both places.)

Example 1:

```
msg text "Hello, $n"
msg send
msg text The current time is $datestr()
msg send
```

Example 2:

```
msg text User $n logged in at $datestr()
log msgs
msg clear
msg text User's break-enabled state is $val(u\$p\BreakTimer\Enabled,1)
log msgs
msg clear
```

---

nop – performs no function. A placeholder that can be useful in certain places.

Syntax options:

nop

Notes:

Do nothing. Can be used, e.g. in conditional statement, to indicate that nothing should be done for a conditional result.

---

noerr – performs a command without reporting errors

Syntax options:

noerr RSIScriptCommand

Notes:

Allows you to execute commands that don't report errors if the commands fail or don't exist (e.g. if you use a command that requires a later version of RSIGuard).

**notice – show a popup Notification that can trigger an RSIScript

Syntax options:

notice msg BORDER-COLOR TIMEOUT/auto TITLE MESSAGE RSISCRIPT
notice msgtimeoutcmd RSISCRIPT
notice width/height [DIMENSION] – select an alternative width/height to the default (if omitted, use default)
notice logo noshow/center [FADELEVEL]/topleft – determine if organization logo should appear and where
notice iconcmd sticky/once CMD – associate an RSIScript command with a click on the RSIGuard system tray icon
notice iconcmd clear – clears the association created by a "notice iconcmd sticky/once" command
notice iconmsg MSG – associate a message to show when mouse pauses over system tray icon, and cause icon to blink
notice winstyle 1/0 – if 1, forces RSIGuard to use Windows style notifications (Windows only, limited functionality)

Notes:

This command is only available for OES scripts. BORDER-COLOR uses HTML format (e.g. FFFFFF for white). Timeout is in seconds – a 0 means never timeout, 'auto' means RSIGuard selects a reasonable timeout. "notice logo" determines where logo goes. If logo is 24-bit, then FADELEVEL sets fade percentage for centered logo (range 5-100). Notice iconcmd & iconmsg allow you to change the behavior of the system tray icon. To clear an iconmsg, use the command "notice iconmsg –". "Sticky iconcmd's" work for multiple clicks, "Once iconcmd's" are cleared after clicking. "notice msgtimeoutcmd" lets you define a command that is executed if the subsequent "notice msg" times out. Notifications are not always shown immediately – display follows typical Windows rules for user interruption. However, setting the border to ffffff will cause the notification to bypass these rules and always display immediately.

---

*open – open a file, folder, application or webpage

Syntax options:

open FILENAME
open FOLDER
open APPLICATION COMMANDOPTIONS
open WEBPAGE

Notes:

Open specified item. In a restricted script, this is limited to opening Remedy Interactive webpages.

Example:

#open c:\My Documents\phonelist.txt in default editor for .txt files
open "c:\My Documents\phonelist.txt"

#open c:\My Documents folder in default file browser (e.g. Explorer on Windows)
open "c:\My Documents"

#open Notepad application
open notepad.exe

#open c:\My Documents\phonelist.txt in Notepad
open notepad.exe "c:\My Documents\phonelist.txt"

#open http://www.rsiguard.com website in default browser
open http://www.rsiguard.com

---

*printf – output text to console (Unix/Linux only)

Syntax options:

printf MESSAGE

---

processmsgs – allow threaded operations to occur

Syntax options:

processmsgs

Notes:

This technical command is generally only used by Remedy Interactive or software engineers to allow certain script operations to happen in parallel with other RSIGuard operations.

---

queryset – ask user for the value of a setting

Syntax options:

queryset SETTING [validate EXPR] [caption CAPTION] number/text PROMPT (ask user for the value of SETTING)
queryset SETTING [validate EXPR] [caption CAPTION] number/text PROMPT VAL1 … VAL-N (give N choices)

Notes:

This command is like the 'set' command described above, except that here the user provides the value for the setting. If values are given, then instead of a freeform dialog, a combobox with the provided options is shown to the user. If "validate" is specified, OK box is greyed until EXPR is true. EXPR should be surrounded by {{}} and the currently entered value will replace $!$. For example, to test if the entered value is 8 characters, EXPR would equal {{$equal($len($!$),8)}}. Be careful with "validate" – if you specify an EXPR that can't be met, the window can't be closed without killing the RSIGuard process (e.g. from Task Manager, or by logging out).

Example:

```
queryset m\RSIGuard\DataFolder text "Enter the network path for the data folder:"
queryset m\RSIGuard\DataFolder text "Enter the network path for the data folder:" "S:\Group1\Data" "S:\Group2\Data"
queryset m\Custom\Value number "How many different computers do you use?"
queryset u\GeneralSetup\EmployeeID validate {{$and($equal($len($!$),8),$isdigits($!$))}} text "Enter 8 digit EEID"

#survey example with results written to text file
log file S:\Network\Survey.txt
queryset u\SurveyQuery\Response text "What is your level of satisfaction?" High Moderate Low
#save response to log file
msg text User $u satisfaction level is: $val(u\SurveyQuery\Response)
msg logclr
queryset u\SurveyQuery\Response text "In a few words, why did you answer $val(u\SurveyQuery\Response)?"
#save response to log file
msg text User $u explanation is: $val(u\SurveyQuery\Response)
msg logclr

#survey that compares estimated work time with measured computer usage, and writes results to text file
queryset u\SurveyQuery\Response validate {{$between(0.1,$!$,99)}} number "How many hours/week do you work?"
#note user's work estimate as well as measured usage
msg text User $u's perceived weekly hours:$val(u\SurveyQuery\Response) vs. actual weekly hours:$data(TH,7,1)
msg logclr
```

See also: question, $choice()

---

question – ask the user a question, and execute a command based on the answer

Syntax options:

question prompt PROMPT (sets the question to be asked)
question caption CAPTION (sets the caption of the popup window)
question answer ANSWER (adds another possible answer to question)
question ask CMD1 [CMD2…] (asks the question and executes associated command(s)). Question text is centered.
question askl CMD1 [CMD2…] (asks the question and executes associated command(s)). Question text is left-justified.

Notes:

Allows conditional execution of a command based on a user's answer to a question. Up to 5 answers can be given, and the number of specified CMDs in the "question ask" command must correspond to the number of "question answer *" commands issued. Using \n in the PROMPT creates a line-break.

Example:

```
question prompt Which division are you in?
question answer "Laser research"
question answer "Materials processing"
question answer "Data analysis"
question ask "read c:\laser.txt" "read c:\materials.txt" "read http://www.acme.com/analysis.txt"

question prompt Visit which website?
question caption Website Decision…
question answer "RSIGuard"
question answer "Google"
question ask "open http://www.rsiguard.com" "open http://www.google.com"

question prompt Thank you for using RSIGuard.\n\nPlease specify your office location?
question caption Location
question answer California
question answer New York
question ask "msg text The Helpdesk is at x1234;msg send" "msg text The Helpdesk is at x5678;msg send"

#survey
log file S:\Network\Survey.txt
msg text "User $u selected: "
question prompt What's your favorite color?
question caption Color Survey
question answer Red
question answer Green
question answer Blue
question ask "msg text Red" "msg text Green" "msg text Blue"
msg logclr
```

See also: queryset, $choice()

---

read – execute a script file

Syntax options:

read FILESPEC (process script file at FILESPEC)

Notes:

Execute commands in a script file. Calls into script files can be nested arbitrarily deeply. FILESPEC can be either a local filename, a network filename, or an http:// based URL. Not available to insecure scripts. OES scripts can only execute scripts from trusted websites.

Example:

```
read c:\RSIGuard Scripts\doit.txt
read http://www.yourcompany.com/rsiscripts/doit.txt
```

---

refresh – refresh the user interface state (buttons, menus, etc.)

Syntax options:

refresh

Notes:

After some script commands (e.g. set), the user interface may need to be updated. For example, if you enable/disable a menu item or if you turn AutoClick/BreakTimer/ForgetMeNots on/off, the user-interface must be updated to reflect this. Use the refresh command after such operations so that the user interface is immediately updated.

---

register – register RSIGuard using a specified registration code

Syntax options:

register REGISTRATIONCODE (registers user if they are not currently registered)

Notes:

If users run RSIGuard by loading RSIGuard off the server every time they start up, then only the copy of RSIGuard on the server needs to be registered with a registration code (and thus you don't need to use this command). However, if each user is installing RSIGuard on their personal machine, each copy of RSIGuard must be individually registered. This command allows the registration process to be automated. If a user is not yet registered, then this command will register their copy using REGISTRATIONCODE (assuming it is a valid registration code). If the registration code starts with 'x', then internet access is required to successfully complete the registration process. 'register 0' unregisters a copy of RSIGuard.

Example:

register xacme1234

***restart** – restarts RSIGuard

Syntax options:

restart

Notes:

Exits RSIGuard and restarts (saving all temporary data and performing and extended form of the refresh command).

---

return – stop execution of a script

Syntax options:

return

Notes:

Generally used in a conditional statement ($if) to terminate processing of a script if a specified condition exists. If running in a nested script, execution in parent script continues.

See also: exit

---

schedule – specify a command to execute at a particular time

Syntax options:

schedule [id ID] in SECONDS CMD
schedule [id ID] at HHMMSS CMD
schedule [id ID] atstamp TIMESTAMP CMD
schedule [id ID] while CONDITION each PERIOD max MAXREPETITIONS CMD
schedule stop IDPREFIX
schedule show

Notes:

Queue commands for later execution. An optional ID can be assigned to any queued command.

The command CMD will execute at the specified delay. Up to 24 commands can be queued at once.

"schedule in" lets you specify a number of seconds that should pass before executing the command. "schedule at" lets you specify a time within 24 hours when the command should execute (e.g. 164000 would be at 4:40PM). If the time specified has passed, the event is scheduled for the following day. "schedule atstamp" lets you specify the a timestamp (roughly seconds since 1/1/70) at which the command should execute (but the event time must be between now and 24 hours from now).

"schedule while" lets you set a command to repeatedly occur while the condition is true (generally must be wrapped in {{}} to delay processing). The command repeats each PERIOD seconds until either the CONDITION is false, or the command has repeated MAXREPETITIONS times.

"schedule stop" will stop any queued commands with an ID that starts with IDPREFIX. If IDPREFIX is omitted, then all scheduled commands are stopped.

"schedule show" will show a popup list of any scheduled events.

Note that scheduled commands that don't execute before RSIGuard is restarted will execute after restart.

***scroll** – scroll up or down (simulating scroll wheel)

Syntax options:

Scroll [up/down] [NUMLINES]

Notes:

Scroll (e.g., in text, up or down (down if not specified) by NUMLINES (or 1 if not specified). Usually used in a hotkey.

****sendconfig / senddata** – transmit configuration and DataLogger data to OES database

Notes:

These commands are only for use by the OES to request data from RSIGuard.

---

sendhsr – trigger the submission of a health status report

Syntax options:

sendhsr query/noquery//asktostart

Notes:

The 'sendhsr' command causes RSIGuard to trigger the submission of a Status Report to a previously configured location. 'sendhsr noquery' causes the most recent set of discomfort survey responses to be submitted (even though survey responses may be old or may never have been set). 'sendhsr query' triggers a new discomfort survey window to appear and submits the survey information. 'sendhsr query' is equivalent to a "Health Status Report" (HSR). 'sendhsr asktostart' asks the user if they want to submit an HSR, and if so, begins an HSR survey.

---

set – remotely control RSIGuard settings

Syntax options:

set SETTING VALUE (set the value of the RSIGuard setting specified by SETTING to VALUE)

Notes:

This command lets you control user settings remotely. You specify the setting you wish to change with “SETTING” and what you want to set it to with “VALUE”. Each setting that appears in the RSIGuard Settings Dialog (and several others) can be specified with the SETTING argument. The key is to understand how SETTING is specified. Each RSIGuard setting has a location within the registry, and SETTING must correspond to that location.

All changeable settings are stored either in HKEY_LOCAL_MACHINE\Software\RSIGuard\Settings or HKEY_CURRENT_USER\Software\RSIGuard\Settings. The SETTING identifier begins with a m\ for the HKEY_LOCAL_MACHINE settings (i.e. m for machine-based settings) and u\ for the HKEY_CURRENT_USER settings (i.e. user-based settings). The rest of the string is the rest of the registry path to the setting. For example, the organization name is stored in HKEY_LOCAL_MACHINE\Software\RSIGuard\Settings\RSIGuard\OrganizationName, so the SETTING identifier is m\RSIGuard\OrganizationName. To change it, you could use:

```
set m\RSIGuard\OrganizationName ACME Inc.
```

Because most (but not all) HKEY_CURRENT_USER settings are specific to a particular profile, you must also specify the profile you wish to change. Often in a startup file, however, you don’t know the profile – in other words, you want multiple users to use the same startup script and have it affect their current profile. You can specify the current profile with the $p substitution (see pre-processing section). For example, to enable the BreakTimer for the current user/profile, you could use the command:

```
set u\$p\BreakTimer\Enabled 1
```

Note that for boolean settings (settings than can only be true or false), you use 1 for true and 0 for false.

You could also create a command that only changes the setting for a particular profile. For example:

```
set u\_JohnSmith\BreakTimer\Enabled 1
```

The 'set' command is powerful in that it lets you arbitrarily write into a user's profile. There is no checking that the setting you are trying to change is a valid setting or that you are not changing an inappropriate setting. For example, if you misspell something in SETTING, you will create a new (probably irrelevant) setting. If you were to arbitrarily change a setting like "the number of user profiles on this PC", you would likely damage the integrity of the user profiles on the PC.

Examples:

```
# specify where all users's usage data should be stored
set m\RSIGuard\DataFolder \\NETDRIVE\RSIGuard\Data

# specify where Health Status Reports should be filed
set m\RSIGuard\ReportFilingLocation *N:\\NETDRIVE\RSIGuard\HSR

# specify your organization's name
set m\RSIGuard\OrganizationName Acme, Inc.

# This changes the minimum time between breaks for the current profile to 15 minutes
set u\$p\BreakTimer\MinInterBreakTime 15
set u\$p\BreakTimer\MinInterBreakTimeEnabled 1

# This disables access to the ForgetMeNots settings from the user interface
set u\GeneralSetup\AccessFMN 0
```

See also: alertset, fset, queryset

***setenv** – set an environment variable

Syntax options:

setenv ENVIRONMENT-VARIABLE [VALUE]

Notes:

This command sets a system environment variable. The value of environment variables can be accessed with $env()

Example:

setenv RSI_DEPARTMENT_CODE 3
setenv RSI_DEPARTMENT $plus($env(RSI_DEPARTMENT_CODE),1) #increment variable

---

setprofile – selects a settings profile created through UI or addprofile

Syntax options:

setprofile # shows list of profile names
setprofile PARTOFPROFILENAME # switches to first profile whose name contains PARTOFPROFILENAME

Notes:

This command lets you move between a set of RSIGuard configuration settings. See *addprofile*

---

setvar – creates an alias for a string to be used elsewhere in script

Syntax options:

setvar 1-9 VALUE

Notes:

Defines 1 of 9 string variables that can be substituted in subsequent commands by a '$' followed by the variable number.

Example:

setvar 1 \\networkdrive\rsiguard
set m\RSIGuard\DataFolder $1\data
set m\RSIGuard\StartupFile $1\scripts\startup.txt
set m\RSIGuard\NewUserScript $1\scripts\newuser.txt

---

speak – speaks text using Text-To-Speech (TTS) engine

Syntax options:

speak text TEXT-TO-SPEAK (*speak the specified text*)
speak selected (*speak the text in the copy buffer*)
speak stop (*stop speaking*)

Notes:

Available only in RSIGuard for Windows, English version. Command is ignored for other platforms.

***,**stretchedit** – edits the video rotation list for BreakTimer breaks

Syntax options:

stretchedit add end/rand/IDX YTVID ID sit/stand/prone/na [POSSIBLE ADD'L FIELDS]
stretchedit add end/rand/IDX YTCHAN USER latest/random/loop MAXLEN
stretchedit delete ytchan CHANNAME (delete all occurrences of CHANNAME)
stretchedit delete ytvid VIDEOID (delete all occurrences of VIDEOID)
stretchedit delete builtin INDEX (delete a builtin stretch video by it's #)
stretchedit delete item INDEX (delete the n'th entry in the stretch rotation)

Notes:

Adds or deletes items from the video rotation list (as shown in the stretches tab of the RSIGuard settings).

***type** – type specified text

Syntax options:

type TEXT

Notes:

Types specified text as if you typed it from the keyboard. You can type special characters using a keyboard escape. For example 'type <tab>' presses the Tab key. A list of all escaped keys is at http://www.rsiguard.com/help/helpkcchars.htm.

You can also type keys directly using keycodes using <#ddd> where ddd is the keycode in base 10. For example, <#186> gives a semicolon. https://msdn.microsoft.com/en-us/library/windows/desktop/dd375731(v=vs.85).aspx shows codes in hexadecimal for all keys.

You can create KeyControl hotkeys and use text/escapes to make the associated text be typed when the hotkey is pressed.

Example:

```
# type an address
type Jacque Smith<ENTER>123 Main St.<enter>Anytown, CA 90001<enter>

# complete fields in a form
type Value 1<tab>Value 2<tab>Value 3

# type I (meaning me) am here; therefore not there.
type I <shift9>meaning me<shift0> am here<#186> therefore not there.
```

***uninstall** – uninstall RSIGuard

Syntax options:

uninstall VERIFY/NOVERIFY

Notes:

Causes RSIGuard to be uninstalled from the computer. If VERIFY is specified, the user is asked to confirm uninstallation. If NOVERIFY, then uninstall proceeds immediately.

***update** – change folder for DataLogger or Profile data storage and copy data to new folder

Syntax options:

update datafolder/profilefolder NEWFOLDER

Notes:

Change where RSIGuard stores DataLogger data or settings data (i.e. roaming profile). This setting could be changed with the 'set' command, but this command also copies the current data to the new location, providing a mechanism for a seamless transfer of storage location (whereas the set command would just change where data was written in the future).

Example:

```
update datafolder S:\NetDrive\RSIGuard\Data
update profilefolder \\svr010\NetDrive\RSIGuard\Data
update datafolder DEFAULT (use keyword DEFAULT (all uppercase) to set RSIGuard to only store data locally)
update profilefolder (leave off NEWFOLDER to disconnect from roaming profile and store settings only locally)
```

---

validate – ask user if script may take control of computer (i.e. use privileged commands)

Syntax options:

validate

Notes:

Prompts user with warning message that script is attempting to gain full access to the computer. More precisely, it means the script wants to use privileged commands (commands marked with an * in this document). Warns user not to continue unless they trust the script and know the source of the script. This command can't be used by scripts that are sent to RSIGuard via the OES.

---

wait – wait specified time before continuing to run script

Syntax options:

wait SECONDS [while CONDITION]

Notes:

Pauses execution of the script file for the specified number of seconds (within a range of 0.1 to 60 seconds). If the optional 'while' clause is specified, then execution is paused for SECONDS while the CONDITION remains true. This is useful for waiting only as long as is necessary to continue processing. For example, if you have a script that launches a web page, and you need to wait for the web page to load before you continue processing, you could have a while clause that waits for the foreground window to be the appropriate page.

Example:

wait 5 (wait 5 seconds)
wait 1.5 (wait 1 and a half seconds)

#login into ACME website, #open login page
open http://www.fake-acme-website.com/login
#wait up to 10 seconds for "ACME Login" website to load
wait 10 while {{$not($foreground(ACME Login))}}
#even though webpage title is there, it could be a moment before the page is rendered and ready for typing
wait 1
#tab to username field, type username, tab to password field, type password and enter to log in
type <tab>myusername<tab>mypassword<enter>

---

while – process a command while an expression is true (loop)

Syntax options:

while EXPRESSION COMMAND

Notes:

Perform COMMAND until EXPRESSION is false (equal to 0). The expression must be pre-processor escaped or it will only be evaluated once. If the COMMAND is more than 1 command, it must be an escaped $do().

Example:

this example calls loop.txt with general variable $1 set to 5, 4, 3, 2, and 1
setvar 1 5
while {{$greaterthan($1,0)}} read loop.txt

this example displays the numbers 5 down to 1
setvar 1 5
while {{$greaterthan($1,0)}} {{$do(setvar 1 $minus($1,1);msg text $1;msg send)}}

---

window – control aspects of the main RSIGuard display window

Syntax options:

window hide/show/center
window alwaysontop/strainbars/autohidebuttons/showtimes 1/0
window next

Notes:

Affect the display of the main window. “Window Hide” and “Window Show” cause the main window to either hide in the system tray or be visible on the screen. “Window Center” forces the window to be visible and appear at the center of the screen. “Window AlwaysOnTop” sets if the display window should always be on top of other windows. “Windows StrainBar” sets if the strain indicators should be shown in the main display. “Windows AutoHideButtons” sets if the main control buttons should automatically hide when the mouse is not over the RSIGuard window. “Windows ShowTimes” sets if the work times (e.g. time since last break, time to next break, etc.) should appear in the main window. The “windows next” command is similar to Alt-Tab – it cycles focus to the next window in the windows stack.

Examples:

window hide (hide the main RSIGuard window)
window strainbars 1 (show the strainbars)

---

wizard – launch the RSIGuard startup wizard

Syntax options:

wizard

Notes:

This command is used when RSIGuard has been in a silent mode and is being activated – thus it is useful to have users go through the setup wizard.