Novell SecureLogin

3.0

www.novell.com

SCRIPT COMMANDS

Novell®

Legal Notices

Novell, Inc. makes no representations or warranties with respect to the contents or use of this documentation, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to revise this publication and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes.

Further, Novell, Inc. makes no representations or warranties with respect to any software, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to make changes to any and all parts of Novell software, at any time, without any obligation to notify any person or entity of such changes.

You may not export or re-export this product in violation of any applicable laws or regulations including, without limitation, U.S. export regulations or the laws of the country in which you reside.

Copyright © 2002 Novell, Inc. All rights reserved. No part of this publication may be reproduced, photocopied, stored on a retrieval system, or transmitted without the express written consent of the publisher.

Novell, Inc.
1800 South Novell Place
Provo, UT 84606
U.S.A.

www.novell.com

Script Commands
December 10, 2002

Online Documentation: To access the online documentation for this and other Novell products, and to get updates, see www.novell.com/documentation.

Novell Trademarks

eDirectory is a trademark of Novell, Inc. in the United States and other countries.
NDS is a registered trademark of Novell, Inc. in the United States and other countries.
NMAS is a trademark of Novell, Inc. in the United States and other countries.
Novell is a registered trademark of Novell, Inc. in the United States and other countries.
Novell SecretStore is a registered trademark of Novell, Inc. in the United States and other countries.

Third-Party Trademarks

All other third-party trademarks are the property of their respective owners.

Contents

About This Guide

This guide is for network administrators. It provides information on and example scripts for SecureLogin commands.

Additional Documentation

For documentation on understanding, installing, managing, and troubleshooting SecureLogin, as well as information on scripts, see the SecureLogin Administration Guide.

For documentation on terminal emulators, see the Configuration Guide for Terminal Emulators.

For documentation on Novell® SecretStore®, see the SecretStore Administration Guide.

Documentation Updates

For the most recent version of this and other SecureLogin guides, see SecureLogin (http://www.novell.com/documentation) at the Novell Documentation Web page.

Documentation Conventions

In this documentation, a greater than symbol (>) is used to separate actions within a step and items within a cross-reference path.

A trademark symbol (®,™, etc.) denotes a Novell trademark. An asterisk (*) denotes a third-party trademark.

1 Overview

This section provides information on the following

How Information on Commands Is Organized

Commands are listed alphabetically. Information on the commands is presented in table format, as the following table illustrates:

Item	Description
Use with:	Startup scripts, Terminal Launcher, Web, and Windows*
SecureLogin version:	All, 2.5 and later, 3, or 3.0.4
Type:	Action, Dialog Specifier, Flow Control, or Variable Manipulator
Usage:	Command argument / *variable*
Arguments:	argument / *variable* - A brief explanation of the argument or variable
Description:	An explanation of the command and how it is used
Syntax examples:	Examples of the various ways the command can be written in a script
Example:	An example script

SecureLogin Scripting Language

SecureLogin scripts use symbols and variables.

Symbols

The following symbols are often used in scripts. The symbols are not commands.

Symbol	Description
“”	In a script, quotation marks are used around text that is separated by spaces. The quotation marks keep the text together as a group.
#	When placed at the beginning of a line of script, the hash (pound) symbol removes the line that the symbol appears in from the action of the script. The symbol accomplishes the following: ◆ Allows comments or explanations about what is occurring in the script. ◆ Allows a line to be temporarily removed without actually being deleted. Temporarily removing the line can help you troubleshoot the script. ◆ Designates the numbers that follow the # symbol as control IDs or class numbers.

Variables

For the purposes of the script language, a variable is any identifier preceded by one of the characters %, $, !, or ?. This identifier is called the variable prefix. The variable prefix determines the type of the value and the steps that the parser should take to determine the value.

At runtime, the script parser substitutes the variable identifiers with their values. For example, the identifiers %CN, $username, and !default are substituted by the values stored as CN, username, and default.

All variables, including runtime variables, are exclusive to the application that the variable was created to work with. To access these variables from another application, you can do either of the following:

- Use the SetPlat command.
- Specify the application name after the variable name.

For example, ?Username secrem.exe causes the script parser to use the value of the runtime variable ?Username. SecureLogin has saved ?Username with a script to be used with the program secrem.exe.

Variable	Function	Description
%	Directory attribute	The Directory attribute instructs the parser to read a value from the current user's object in the Directory. The attribute must be a string. The attribute is read each time that the script is run.

Variable	Function	Description
!	Pass ticket	The ! character specifies that a one-time password should be generated. To generate a pass ticket, a DES Key and Offset are required. At runtime, the parser looks for the variables DESKEY and DESOFFSET in the application specified as the rest of the variable identifier. For example, the variable !Novell looks for the variables DESKEY and DESOFFSET in the “Novell” application. The special identifier !default reads these variables from the current application. If either DESKEY or DESOFFSET are not found, then random values are generated and saved in the designated application. Each application script has an associated symbol table. A symbol table is referenced by its application name and contains a list of variables and values that can be used in the script.
?	Runtime	A variable prefixed with the question mark character (?) is classified as a runtime variable. The variable is only valid for the life of proto.exe, unless the variable is declared as a “local” variable.
$	Symbol table	A variable prefixed with the $ character is looked up in the current application entry. If the value that was substituted also has a $ character as its first character, it is also run through the substitution engine. This only occurs once. If the symbol is not found in the current application, the user is prompted to enter the variable's value. If the user enters a value, it is substituted immediately and the value is saved.

SecureLogin supports internal variables. The script language can use these variables to

- Determine what operations can be performed
- Extract machine system information or network information about the local user

Variable Name	Description
?*SYSVERSION*	The local SecureLogin Windows agent version. This variable can be used to determine if specific support is built into the product running on the user’s workstation. The format of the variable is major.minor.build (for example, 2030008, which represents v2.3.0.8).
?*BROWSERTYPE* (system)	Contains either Internet Explorer or Netscape* and indicates which browser the script is running in.
?*SYSUSER* (system)	The name of the user that was last used in the GINA or Windows 9*x* login panel. This variable will only be available when the SecureLogin login extension is installed.
?*SYSPASSWORD* (system)	The password that matches the username presented in the GINA panel. This value can only be retrieved if the SecureLogin login extension is installed in the GINA or Windows 9*x* login panel.

Variable Name	Description
?*SYSCONTEXT* (system)	Lists the Novell® NDS® or eDirectory™ user context as entered in the GINA or Windows 9*x* login panel. This variable requires the login extension to be installed.
?*SYSTREE* (system)	The NDS or eDirectory tree name that the user entered. This variable requires the login extension to be installed.
?*SYSSERVER* (system)	The name of the server that was entered in the Login GINA or Windows 9*x* login panel. This variable requires the login extension to be installed.
?CurrTime	Tthe running time in seconds from January 1970 to the present. This variable is used for anything that requires a time measurement (for example, to force password changes every *x* days).

2 SecureLogin Commands

This section provides information on commands used in SecureLogin scripts. The information includes example scripts.

AAVerify

Item	Description
Use with:	Startup scripts, Terminal Launcher, Web, and Windows
SecureLogin version:	3.0.4
Type:	Action
Usage:	AAVerify [-user *user object*] [-tree *tree name*] [-method *nmas sequence*] [*return variable*]
Arguments:	
user object	The DN of the user to be reauthenticated.
tree name	The user's NDS or eDirectory tree name.
nmas sequence	The NMAS login sequence to be used.
return variable	An optional variable that can store the results of the verification. Output values are "true" or "false." If *return variable* is specified, the script continues, regardless of the results of reauthentication. If no variable is specified, the script continues to execute following a successful reauthentication or terminates following an usuccessful reauthentication.
Description:	Used with Novell Modular Authentication Service (NMAS™) to verify the identity of the user before permitting single sign-on to an application. If you don't specify *user object*, *tree name* or, *auth method*, AAVerify defaults to the values used at login. If NMAS is not installed on the workstation, the script sends an error, or an error is returned via *return variable*. To enable AAVerify with NMAS, make sure that nmas.dll is in the PATH. Also make sure that the NMAS client and specified login sequence are installed and properly configured. (See the NMAS documentation for details.)

Item	Description

Example:	`Dialog`
	` Title "Login"`
	` Ctrl #1`
	`EndDialog`
	`AAVerify`
	`Type $Username #301`
	`Type $Password #302`
	`Type \N`

Add

Item	Description
Use with:	Startup scripts, Terminal Launcher, Web, and Windows
SecureLogin version:	3
Type:	Variable Manipulator
Usage:	Add *variable1 variable2* [?result]
Arguments:	
variable1	The first argument, the number that the second argument will be added to. If the optional ?result argument is not passed in, this argument also contains the result of the addition equation. If you use *variable1* without the ?result argument, *variable1* must be a SecureLogin variable (either ?variable1 or $variable1). Otherwise, *variable1* can be any numeric value.
variable2	The second argument, the number added to the first argument in the equation. V*ariable2* can be a SecureLogin variable or a numeric value.
[?result]	Optional. The sum or result of the equation. See “Subtract” on page 49, “Divide” on page 23, and “Multiply” on page 32.

Example:	`Add 1 2 ?res`
	`#?res will now be 3`
	`Set ?res2 4`
	`Add ?res ?res2`
	`#?res will now be 7`

BeginSplashScreen / EndSplashScreen

Item	Description
Use with:	Terminal Launcher (Generic and Advanced Generic only)
SecureLogin version:	3.0.4
Type:	Action

Item	Description
Usage:	BeginSplashScreen EndSplashScreen
Arguments:	None
Description:	Displays a Novell splash screen across the whole Terminal Emulator window. This command is used to mask any flashing, etc. that is produced by SecureLogin selecting text from the screen. A Delay command at the start of the script ensures that the emulator window is in place before the splash screen is displayed.

Example:	`Delay 2000`
	`BeginSplashScreen`
	`WaitForText "ogin:"`
	`EndSplashScreen`
	`Type $Username`
	`Type @E`
	`Begin SplashScreen`
	`WaitForText "assword:"`
	`EndSplashScreen`
	`Type $Password`
	`Type @E`

Break

Item	Description
Use with:	Startup scripts, Terminal Launcher, Web, and Windows
SecureLogin version:	2.5 and later
Type:	Action
Usage:	Break
Arguments:	None
Description:	Used within the Repeat/EndRepeat commands to break out of a repeat loop.

Example:	`Dialog`
	`  Title "Login"`
	`  Ctrl #1`
	`EndDialog`
	`Repeat`
	`  ReadText #301 ?Text`
	`  If ?Text eq Login`
	`  Break`
	`EndIf`
	`EndRepeat`

Call

Item	Description
Use with:	Startup scripts, Terminal Launcher, Web, and Windows
SecureLogin version:	2.5 and later
Type:	Flow Control
Usage:	Call *subroutine*
Arguments:	
subroutine	The name of the subroutine to be called. This name must be identical to the name specified in the Sub command.
Description:	Calls and runs a subroutine.

Example:	`Dialog`
	` Title "Login"`
	` Ctrl #1`
	`EndDialog`
	`If -Text "Login"`
	` Call Login`
	`EndIf`
	`If -Text "Wrong Password"`
	` Call WrongPassword`
	`EndIf`

ChangePassword

Item	Description
Use with:	Startup scripts, Terminal Launcher, Web, and Windows
SecureLogin version:	All
Type:	Action
Usage:	ChangePassword *variable* [-random]
Arguments:	
variable	A normal or runtime variable that the new password is stored in.
[-random]	Invokes the random password generator.
Description:	Allows a single variable to be changed. Use this command in scenarios where password expiration is an issue. The variable will be set to the new password. The flag for this command is -random. If -random is set, the new password will be generated automatically in compliance with the variable's password policy. If -random is not set, a dialog box prompts the user to enter a new password. The new password will be tried against any variable password policies that are in place. See "RestrictVariable" on page 39.

Item	Description
Syntax examples:	ChangePassword ?NewPassword ChangePassword ?NewPassword "Please enter a new password” ChangePassword ?NewPassword random

Example:	`Dialog`
	`  Title "Login"`
	`  Ctrl #1`
	`EndDialog`
	`If -Text "Enter your username and password"`
	`  Type $Username #1`
	`  Type $Password #2`
	`  Click #1`
	`EndIf`
	`If -Text "Enter new password"`
	`  Set $PasswordBackup $Password`
	`  Type $Password #1`
	`  ChangePassword $Password random`
	`  Type $Password #2`
	`  Type $Password #3`
	`  Click #1`
	`EndIf`

Class

Item	Description
Use with:	Windows
SecureLogin version:	All
Type:	Dialog Specifier
Usage:	Class *window-class*
Arguments:	
window-class	A string specifying the window class that this statement will match.
Description:	When a window is created, it is based on a template known as a window class. The CLASS statement checks to see if the class of the newly created window matches its *window-class* argument. If the window matches the *window-class* argument, the execution of the script continues to the next line. If the window does not match the *window-class* argument, execution continues at the next dialog statement. You can determine the class by using Window Finder. See Finding Dialog Control IDs in the *SecureLogin Administration Guide*.

Example:	`Dialog`
	`  Title "Login"`
	`  Class #32770`
	`EndDialog`

Click

Item	Description
Use with:	Web and Windows
SecureLogin version:	All
Type:	Action
Windows Usage:	
Usage 1	Click *#Ctrl-ID* [-rawflag] [-right] [X / Y]
Usage 2	Click *X_coordinate Y_coordinate*
Windows arguments:	
#Ctrl-ID	The ID number of the control to be pressed.
[-raw]	Eliminates the mouse and sends a direct click.
[-right]	Sends a right-mouse click. Use this argument only with the -raw flag.
X_coordinate	Represents the horizontal coordinate relative to the client area of the application (not the screen).
Y_coordinate	Represents the vertical coordinate relative to the client area of the application (not relative to the screen).
Web usage:	
Usage 1	Click *number*
Usage 2	Click *#number*
Web arguments:	
#number	The pound or hash symbol followed by the sequential number of the button to be pressed. The number of the button is determined by the Web page layout. The first button (top to bottom, left to right) on the page is number 1, the second button on the page is number 2, and so on. Because of Web page layout and design, the sequential order of the buttons might not be obvious.
number	The sequential number of the button to be pressed. The number of the button is determined by the Web page layout. The first button on the page is number 1, the second button on the page is number 2, and so on. Because of Web page layout and design, the sequential order of the buttons might not be obvious.

Item	Description
Description:	When used with Windows applications, the Click command sends a click command to the specified *#Ctrl-ID*. If the button to be clicked does not have a control ID, use the Type \n command. The -raw flag causes SecureLogin to bypass the mouse by emulating the mouse and sending a direct click message to the control. Using the -right flag with the -raw flag sends a right-click to the control. The -raw flag can be set if the button or control does not respond to the Click command. Setting the *#Ctrl-ID* to 0 (zero) sends the Click command to the window that the script is running on. *X coordinate Y coordinate* coordinates can also be set. The *X coordinate Y coordinate* coordinate is relative to the client area of the application, not the screen. When used with Web, the click command takes a single argument, which is the sequential number on the page of the button to be pressed. Click #3 clicks the third button on the page. Keep in mind that, because of Web page layout and design, the sequential order of the buttons might not be obvious.
Syntax examples:	Click #1 Click #1 -raw -right Click -x 12 -y 12

Example:	`Dialog`
	`  Title "Login"`
	`  Ctrl #1`
	`EndDialog`
	`Type $Username #1`
	`Type $Password #2`
	`Click #1`

ConvertTime

Item	Description
Use with:	Startup scripts, Terminal Launcher, Web, and Windows
SecureLogin version:	3.0.4
Type:	Variable Manipulator
Usage:	ConvertTime ?Currtime *string time*
Arguments:	
string time	The output variable.
Description:	Converts ?Currtime system to a string representation of the date and time and stores it in *string time*.
Example:	ConvertTime ?Currtime ?Time MessageBox ?Time

Ctrl

Item	Description
Use with:	Windows
SecureLogin version:	All
Type:	Dialog Specifier
Usage:	Ctrl *#Ctrl-ID* [*regex*]
Arguments:	
#Ctrl-ID	The ID number of the control to be checked.
[*regex*]	The regular expression.
Description:	Determines if a window contains the control expressed in the *#Ctrl-ID* argument. The control ID number is a constant that is established at the time a program is compiled. **NOTE:** Third-party software control ID numbers might not be constant from one version to the next. You can use the Window Finder tool to determine the control ID number. See Finding Dialog Control IDs in the *SecureLogin Administration Guide*. Using the [*regex*] argument adds a further check that allows the script to skip to the next command. If the text on the specified *#Ctrl-ID* does not conform to the [*regex*], the script skips to the next dialog statement as though the *#Ctrl-ID* did not exist.
Syntax examples:	Ctrl #1 Ctrl #1 OK

Example:	`Dialog`
	`  Title "Login"`
	`  Ctrl #1 OK`
	`  Ctrl #2 Cancel`
	`  Ctrl #3 Help`
	`EndDialog`
	`Type $Username`
	`Type \T`
	`Click #1`

Delay

Item	Description
Use with:	Startup scripts, Terminal Launcher, Web, and Windows
SecureLogin version:	All
Type:	Action
Usage:	Delay *period-of-time*

Item	Description
Arguments:	
period	A period of time, expressed in milliseconds, to pause execution of the script.
Description:	Delays the action of the script for the time specified in the *period* argument. For example, Delay 5000 creates a 5-second pause. Use the Delay command to accommodate an introduction screen or some other custom feature. When troubleshooting new scripts, add Delay to the script as a first course of action.

Example:	`Dialog`
	`  Title "Login"`
	`  Ctrl #1`
	`EndDialog`
	`Delay 500`
	`Type $Username #1`
	`Type $Password #2`
	`Click #1`

Dialog / EndDialog

Item	Description
Use with:	Windows
SecureLogin version:	All
Type:	Dialog Specifier
Usage:	Dialog EndDialog
Arguments:	None
Description:	Identifies the beginning and end of a dialog specifier block. Use these commands to construct a dialog specifier block, which consists of a series of dialog specifier statements (for example, Ctrl, Title). When a dialog block is executed, each of the dialog specifier statements is executed in sequence. If any statement within the dialog block fails, the entire dialog block fails and execution proceeds to the next Dialog/EndDialog block, if any. The part of the script that follows the EndDialog command is called the script body. Another dialog block, or the end of the script, terminates the script body.

Example:	`Dialog`
	`  Title "Login"`
	`  Ctrl #1`
	`EndDialog`
	`Type $Username #1`
	`Type $Password #2`
	`Click #1`

DisplayVariables

Item	Description
Use with:	Startup scripts, Terminal Launcher, Web, and Windows
SecureLogin version:	All
Type:	Action
Usage:	DisplayVariables [prompt-text]
Arguments:	
Prompt-text	Optional, customized text to be displayed in the Enter SecureLogin Variables dialog box.
Description:	Displays a dialog box that lists the user's variables for the current application. The user can edit the variables from this dialog box. For example, if the login is unsuccessful because of an incorrect username or password, the DisplayVariables command prompts the user to edit the stored username or password values. From that point, the login process proceeds as usual. To replace the default prompt text in the Enter SecureLogin Variables dialog box, enter the replacement text in quotation marks after the DisplayVariables command. If no variables are stored for the user the first time SecureLogin attempts to apply single sign-on to the application, the prompt will not be customized. After variables are stored for the user, the prompt is customized when the script is run. You can also customize the text in the prompt by using the SetPrompt command. See "SetPrompt" on page 44.
Syntax examples:	DisplayVariables DisplayVariables "Please enter your details" DisplayVariables "Please enter a new password" $Password DisplayVariables "Please enter your username and password" $Username $Password DisplayVariables "" $Username $Password

Example:	`Dialog`
	`  Title "Wrong Password"`
	`  Ctrl #1`
	`EndDialog`
	`DisplayVariables "Enter a new username and password"`
	`Type $Username #1`
	`Type $Password #2`
	`Click #1`

Divide

Item	Description
Use with:	Startup scripts, Terminal Launcher, Web, and Windows
SecureLogin version:	3
Type:	Variable Manipulator
Usage:	Divide *variable1 variable2* [?result]
Arguments:	
variable1	The dividend. The first argument. The number that will be divided by the second argument. This argument will contain the result if the optional [?result] argument is not passed in. If you use the *variable1* argument without the [?result] argument, *variable1* must be a SecureLogin variable (either ?variable1 or $variable1). Otherwise, *variable1* can be any numeric value.
variable2	The divisor. The second argument. The number that the first argument is divided by. The *variable2* argument can be a SecureLogin variable or a numeric value.
[?result]	The quotient or result of the equation. See “Add” on page 14, “Subtract” on page 49, and “Multiply” on page 32.

Example:	`Divide 6 2 ?res`
	`#?res will now be 3`
	`Set ?res2 3`
	`Divide ?res ?res2`
	`#?res will now be 1`

EndScript

Item	Description
Use with:	Startup scripts, Terminal Launcher, Web, and Windows
SecureLogin version:	All
Type:	Action
Usage:	EndScript
Arguments:	None
Description:	Immediately terminates execution of the script.

Item	Description

Example:	`Dialog`
	` Title "Login"`
	` Ctrl #1`
	`EndDialog`
	`Type $Username #1`
	`Type $Password #2`
	`Click #1`
	`If -Text "Incorrect Password"`
	` MessageBox "You have entered an incorrect`
	` password"`
	` EndScript`
	`EndIf`

GetCommandline

Item	Description
Use with:	Windows
SecureLogin version:	3.0.4
Type:	Action
Usage:	GetCommandline *variable*
Arguments:	
variable	Defines where the captured command line will be stored.
Description:	Captures the full command line of the program that is loaded, and saves it to the specified *variable*.

Example:	`GetCommandline ?Text`
	`If ?Text eq "C:\Winnt\Notepad.exe`
	` Killapp Notepad.exe`
	`EndIf`

GetText

Item	Description
Use with:	Web components (for example, Internet Explorer and Netscape)
SecureLogin version:	2.5 and later
Type:	Action
Usage:	GetText *variable*

Item	Description
Arguments:	
variable	Defines where the captured text will be stored.
Description:	Gets all of the text from the screen and saves it to the specified *variable*. This command Is rarely used and is generally unnecessary. Use this command in a large Web script that might contain several If-text statements. Under Netscape, each If-text statement scans the screen to find the specified text. Each scan of the screen will result in the screen flashing. However, if you use GetText (for example, if ?text -in ?URLFromGetText), the script can contain multiple If-text commands, with only one scan of the screen.

Example:	`GetText ?Text`
	`If Login -in ?Text`
	`  Type $Username`
	`  Type $Password Password`
	`EndIf`

GetURL

Item	Description
Use with:	Web components (for example, Internet Explorer and Netscape)
SecureLogin version:	2.5 and later
Type:	Action
Usage:	GetURL *variable*
Arguments:	
variable	Defines where the captured URL will be stored.
Description:	Captures the URL of the site that is loaded and saves it to the specified *variable*.

Example:	`GetURL ?URL`
	`If ?URL eq "http://www.yahoo.com"`
	`  Type $Username`
	`  Type $Password Password`
	`EndIf`

GotoURL

Item	Description
Use with:	Web components (for example, Internet Explorer and Netscape)
SecureLogin version:	2.5 and later
Type:	Action

Item	Description
Usage:	GotoURL *URL*
Arguments:	
URL	The URL that the browser will navigate to.
Description:	Makes the browser navigate to the specified *URL*.

Example:	`If -text "Incorrect Password"`
	` MessageBox "You have entered an incorrect`
	` password"`
	` GotoURL "http://www.novell.com"`
	`EndIf`

If / Else / EndIf

Item	Description
Use with:	Startup scripts, Terminal Launcher, Web, and Windows
SecureLogin version:	All
Type:	Flow Control

Usage 1:	If *value1 Gt/Lt/Eq* v*alue2*
	Else
	Endif

Usage 2:	If -text *text*
	Else
	EndIf

Usage 3:	If -exists *variable*
	Else
	EndIf

Usage 4:	If *text to find* -in *text_to_search_through*
	Else
	EndIf

Item	Description
Arguments:	
Gt/Lt/Eq/-in	Operators. They compare the value on the left side of the operator to the value on the right side of the operator. ◆ If *value1* eq *value2* Assesses whether two values are equal. ◆ If *value1* gt *value2* Assesses whether one value is greater than another value. ◆ If *value1* lt *value2* Assesses whether one value is less than another value. ◆ If *text_to_find* -in *text_to_search_through* Searches for text from within specified text.
-text	If text *text*. A shortcut to evaluate text within windows.
-exists	If -exists *variable*. Assesses whether the text specified in the *variable* is present.
Description:	Establishes a block to be executed if the operator is found to be true. The Else command works inside an If block. This command is executed if the operator in the If block is false. The EndIf command terminates the If block.
Syntax examples:	If ?Value1 gt ?Value2 If -Text "Login” If -Exists ?RunBefore If Login -in ?Text

Example:	`If -text "Incorrect Password"`
	`MessageBox "You have entered the incorrect password"`
	`GotoURL "http://www.novell.com"`
	`Else`
	`MessageBox "You have been successfully logged in"`
	`EndIf`

Include

Item	Description
Use with:	Startup scripts, Terminal Launcher, Web, and Windows
SecureLogin version:	3.0 and later
Type:	Flow Control
Usage:	Include [*platform-name*]
Arguments:	
[*platform-name*]	The name of the script to be included..

Item	Description
Description:	Allows commonly-used application script code to be shared by multiple applications. The script identified by *platform-name* is included at execution time into the calling application script. The application type selected for the script to be included should be compatible with the calling script's application type.
Syntax examples:	Include HelloLib
Example:	```## "notepad.exe" application script Include HelloLib Dialog Title "Untitled - Notepad" EndDialog Type abc Call printHelloMsg Dialog Title "Notepad" Ctrl #6 EndDialog Call printGoodbye Click #7 #"HelloLib" platform script Sub printHelloMsg MessageBox "Hello Message" EndSub Sub printGoodbye MessageBox "Goodbye" EndSub```

Increment / Decrement

Item	Description
Use with:	Startup scripts, Terminal Launcher, Web, and Windows
SecureLogin version:	All
Type:	Flow Control
Usage:	Increment *variable* Decrement *variable*
Arguments:	
variable	The name of the variable used to control the count process.

Item	Description
Description:	Counts the number of passes a particular script has made. After the number of instances is equal to the specified number, you can instruct the script to run another task or end the script. This instruction can be particularly useful in the following situations: ◆ When you configure an application whose login panel is similar to other windows within the application. ◆ To easily control the number of attempts a user can have to access an application.
Syntax examples:	Increment ?RunCount Decrement ?RunCount

Example:	`Dialog`
	`  Title "Login"`
	`  Ctrl #1`
	`EndDialog`
	`Increment ?RunCount`
	`If ?RunCount gt 5`
	`  MessageBox "You have attempted too many logins. The session will be terminated."`
	`  KillApp login.exe`
	`EndIf`

KillApp

Item	Description
Use with:	Startup scripts, Terminal Launcher, Web, and Windows
SecureLogin version:	All
Type:	Flow Control
Usage:	KillApp *process-name*
Arguments:	
process-name	The name of the process that will be terminated.
Description:	Terminates a process.

Example:	`Dialog`
	`  Title "Login"`
	`  Ctrl #1`
	`EndDialog`
	`Increment ?RunCount`
	`If ?RunCount gt 5`
	`  MessageBox "You have attempted too many logins. The session will be terminated."`
	`  KillApp login.exe`
	`EndIf`

Local

Item	Description
Use with:	Startup scripts, Terminal Launcher, Web, and Windows
SecureLogin version:	3
Type:	Variable Manipulator
Usage:	Local *?variable*
Arguments:	
?variable	The runtime variable that will be declared as local.
Description:	Declares that a runtime variable will only exist for the lifetime of the script. Use local runtime variables the same way as normal runtime variables, and still write local runtime variables as ?variable. Declare local runtime variables to be local by using the Local command, followed by the variable name. When runtime variables are declared local, they cannot be set back again. You can declare a runtime variable local at any time in a script. Using local runtime variables slightly increases the performance of SecureLogin. Use Local runtime variables to run scripts multiple times and not have the runtime variables stored between each run of the script. Also use local runtime variables to prevent runtime variables from overwriting each other. Overwriting could happen if two instances of a script are running at the same time. For example, use the Local command if two instances of Terminal Launcher are running, each instance running the same script but attached to different emulator sessions.

Example:	`Dialog`
	`  Title "Login"`
	`  Ctrl #1`
	`EndDialog`
	`Local ?RunCount`
	`Increment ?RunCount`
	`If ?RunCount gt 5`
	`  MessageBox "Closing Application"`
	`  KillApp login.exe`
	`EndIf`
	`Type $Username`
	`Type $Password Password`

MessageBox

Item	Description
Use with:	Startup scripts, Terminal Launcher, Web, and Windows
SecureLogin version:	All
Type:	Action

Item	Description
Usage:	MessageBox [-yesno flag] [-yesnocancel flag] [-background] *?variable* [“-defaultno”] *data*
Arguments:	
[-yesno]	Allows the user to select either Yes or No within the message box rather than being limited only to an OK button.
[-yesnocancel]	Allows the user to select either Yes, No, or Cancel when a message box is presented.
[-background]	When specified, -background allows the user to open an application and work in that application, without responding to the MessageBox. If this parameter is not used, the MessageBox remains the topmost window and the user must respond to the MessageBox before continuing with any other work.
?variable	This runtime variable is required with the -yesno or -yesnocancel flag to produce a result for the -yesno or -yesnocancel flag. It is only used with the -yesno or -yesnocancel flag.
[-defaultno]	An optional parameter, used only with the -yesno flag. When the -defaultno parameter is set, default focus goes to the No button instead of to the Yes button.
data	The text to be displayed to the user.
Description:	Displays a dialog box that contains the text specified in the *data* variable. The script is suspended until the user reacts to this message. As the following line illustrates, MessageBox can take any number of text arguments, including variables: `MessageBox "The User "$Username" has just been logged into the system”` The -yesno flag can be set when calling a MessageBox. If this flag is set, the MessageBox prompts the user with a box that has a Yes and a No button rather then an OK button. You can capture the result of the MessageBox immediately after the flag by using a runtime *?variable*. The following example sets the value of ?result to Yes if the user clicked Yes or to No if the user clicked No.
Syntax examples:	MessageBox "Script completed successfully” MessageBox "Do you want to continue?” -yesno ?Result MessageBox "Do you want to continue?” -yesnocancel -background -defaultno ?Result

Example:	`Dialog`
	`  Title "Login"`
	`  Ctrl #1`
	`EndDialog`
	`Type $Username #1`
	`Type $Password #2`
	`Click #1`
	`MessageBox "Script completed successfully"`

Multiply

Item	Description
Use with:	Startup scripts, Terminal Launcher, Web, and Windows
SecureLogin version:	3
Type:	Variable Manipulator
Usage:	Multiply *variable1 variable2* [?result]
Arguments:	
variable1	The multiplicand. The first argument. The number that will be multiplied by the second argument. This argument will contain the result if the optional [?result] argument is not passed in. If you use the *variable1* argument without the [?result] argument, *variable1* must be a SecureLogin variable (either ?variable1 or $variable1). Otherwise, *variable1* can be any numeric value.
variable2	The multiplier. The second argument. The number that the first number will be multiplied by. *Variable2* can be a SecureLogin variable or a numeric value.
[?result]	Optional. The product or result of the equation. See “Add” on page 14, “Subtract” on page 49, and “Divide” on page 23.

Example:	`Multiply 6 2 ?res`
	`#?res will now be 12`
	`Set ?res2 3`
	`Multiply ?res ?res2`
	`#?res will now be 36`

OnException/ClearException

Item	Description
Use with:	Startup scripts, Terminal Launcher, Web, and Windows
SecureLogin version:	3.0.4
Type:	Flow Control
Usage:	OnException *exception name* call *subroutine* ClearException *exception name*

Item	Description
Arguments:	
exception name	The name of the exception that you want to act on. Currently two exceptions are supported: ◆ ChangePasswordCancelled When a user clicks Cancel in the Change Password dialog box. ◆ EnterVariablesCancelled When a user clicks Cancel in the automatic variable prompt dialog box.
subroutine	The name of the subroutine you want to run when the exception condition is found to be true.
Description:	Detects when certain conditions are met. Currently, this is when the Cancel button is clicked in either of two dialog boxes. When the condition is met, a subroutine is run. Use the ClearException command to reset the exceptions value.

Example:	`Dialog`
	`  Title "Login"`
	`EndDialog`
	`OnException EnterVariablesCancelled Call VariablesCancelled`
	`DisplayVariables "Please ensure your username and password are correct"`
	`ClearException EnterVariablesCancelled`
	`Type $Username #1`
	`Type $Password #2`
	`Click #1`
	
	`Sub VariablesCancelled`
	`  Messagebox "You cannot cancel the variable verification dialog box. Please make any necessary changes, and click OK"`
	`OnException EnterVariablesCancelled Call VariablesCancelled`
	`DisplayVariables "Please ensure your username and password are correct"`
	`ClearException EnterVariablesCancelled`
	`EndSub`

Parent / EndParent

Item	Description
Use with:	Windows
SecureLogin version:	All
Type:	Dialog Specifier
Usage:	Parent/EndParent
Arguments:	None

Item	Description
Description:	The Parent command begins a parent block in which the statements act upon a window's parent. The commands that follow the Parent command function identically to commands used in a dialog block. If they evaluate to False, the script ends. For example, the command Title in a parent block returns False if the title of the parent does not match the one specified in the command. However, if a command in a parent block returns a False result, the execution does not skip to the next parent block, as it would in a dialog block. Instead, the parent block proceeds to the next dialog box, or the script terminates if no dialog block exists. The EndParent command terminates a parent block and sets the subject of the script back to the original window. You can nest the Parent command, allowing the parent block to act on the parent of the parent.

Example:	`Dialog`
	`Parent`
	`Parent`
	`Title "Login"`
	`Ctrl #1`
	`EndParent`
	`EndParent`
	`EndDialog`

PickListAdd

Item	Description
Use with:	Startup scripts, Terminal Launcher, Web, and Windows
SecureLogin version:	All
Type:	Action
Usage:	PickListAdd *display-text* [*return-value*]
Arguments:	
display-text	The text that will be displayed in the pick list for the specified option.
[return-value]]	The value returned from the pick list. If you don't specify this parameter, the return value will be the display text.
Description:	Allows users who have multiple accounts for a particular system to choose the account that they will log in to. Also, you can use this command to choose from multiple sessions on one mainframe account.

Item	Description

Example:	`Dialog`
	` Title "Login"`
	` Ctrl #1`
	`EndDialog`
	`PickListAdd "Account One" One`
	`PickListAdd "Account Two" Two`
	`PickListAdd "Account Three" Three`
	`PickListDisplay ?Account "Please select the account you want to use" -noedit`
	`SetPlat ?Account`
	`Type $Username #1`
	`Type $Password #2`
	`Click #1`

PickListDisplay

Item	Description
Use with:	Startup scripts, Terminal Launcher, Web, and Windows
SecureLogin version:	All
Type:	Action
Usage:	PickListDisplay *?variable display-tex*t [-noedit]
Arguments:	
?variable	The output variable for the selected option.
*display-tex*t	The description text for the pick list box.
[-noedit]	Prevents the user from adding additional variables.
Description:	Displays the pick list entries built by previous calls to PickListAdd. The PickListDisplay command returns the result in a *?variable* sent to the command. If the desired entry is not among the displayed entries, users can enter their own data into an edit field at the bottom of the pick list. As the following line illustrates, you can turn off this feature by setting the noedit flag: PickListDisplay ?result "Please pick your user” noedit
Syntax examples:	PickListDisplay ?Account PickListDisplay ?Account "Please select the account you wish to use” PickListDisplay ?Account "Please select the account you wish to use” -noedit

Item	Description

Example:	`Dialog`
	` Title "Login"`
	` Ctrl #1`
	`EndDialog`
	`PickListAdd "Account One" One`
	`PickListAdd "Account Two" Two`
	`PickListAdd "Account Three" Three`
	`PickListDisplay ?Account "Please select the account you want to use" -noedit`
	`SetPlat ?Account`
	`Type $Username #1`
	`Type $Password #2`
	`Click #1`

ReadText

Item	Description
Use with:	Terminal Launcher, Windows
SecureLogin version:	All
Type:	Action
Windows Usage:	ReadText #Ctrl-*ID variable*
Terminal Launcher Usage:	ReadText *variable character-number row-number* [column number]
Arguments:	
#Ctrl-ID	The control ID number of the text to be read.
variable	The variable that will receive the text that is read.
character-number	The number of characters to be read.
row number	The horizontal position number of the first character to be read (for example, row).
[column-number]	Optional. The vertical position number of the first character to be read (for example, column). If you don't specify the column-number, the *row-number* is used as the number that the reading will start from. If *column-number* is specified, a row/column conversion is carried out before the reading starts.

Item	Description
Description:	Runs in both Windows and Terminal Launcher scripts. Although the usage and arguments for the use of ReadText with Windows and Terminal Launcher are different, the results of each command are the same. In a Windows script, the ReadText command reads the text from any given *#Ctrl-ID* and sends it to the specified variable. For this command to function correctly, the *#Ctrl-ID* must be valid. In a Terminal Launcher script, the ReadText command reads a specified number of characters, starting at the *row-number*, and sends those characters to the specified *variable*. The ReadText command won't work with Generic or Advanced Generic emulators. It only works with HLLAPI and some DDE emulators.
Syntax examples:	ReadText #301 ?Text ReadText ?Text 4 6

Example:	`Dialog`
	`  Title "Login"`
	`  Ctrl #1`
	`EndDialog`
	`ReadText #301 ?Text`
	`MessageBox ?Text`

RegSplit

Item	Description
Use with:	Startup scripts, Terminal Launcher, Web, and Windows
SecureLogin version:	All
Type:	Action
Usage:	RegSplit *regex_input-strin*g o*utput-string1 output-string2*
Arguments:	
regex	The regular expression.
*input-strin*g	The string that will be split.
o*utput-string1*	The first subexpression.
output-string2	The second subexpression.
Description:	The RegSplit command enables you to split a string by using a regular expression. *Output-string1* contains the first subexpression. *Output-string2* contains the second subexpression.

Item	Description

Example:	`Dialog`
	` Title "Login"`
	` Ctrl #1`
	`EndDialog`
	`ReadText #301 ?Text`
	`Regsplit "Please enter the password for (.*) account" ?Text ?User`
	`SetPlat ?User`
	`Type $Username #1`
	`Type $Password #2`
	`Click #1`

Open text example:	#?InputString: "`This is a long string with a few components in it`"
Command:	Regsplit "`This (.*) a long (.*) with (.*) components (.*)`" ?InputString ?first ?second ?third ?fourth
Result:	?first = "is", ?second = "string", ?third = "a few", ?fourth = "in it"

Repeat / EndRepeat

Item	Description
Use with:	Startup scripts, Terminal Launcher, Web, and Windows
SecureLogin version:	All
Type:	Action
Usage:	Repeat [loop#] EndRepeat
Arguments:	
[loop#]	The number of times the repeat script block is repeated. If you don't specify a number, the repeat continues indefinitely.
Description:	The Repeat command establishes a script block similar to the If command. The EndRepeat command terminates the repeat block. To break out of a repeat block, use the Break command or the EndScript command.
Syntax examples:	Repeat Repeat 3

Example:	`Dialog`
	` Title "Login"`
	` Ctrl #1`
	`EndDialog`
	`Repeat`
	` Type \N`
	` If -Text "Login Successful"`
	`  Break`
	` EndIf`
	`EndRepeat`

RestrictVariable

Item	Description
Use with:	Startup scripts, Terminal Launcher, Web, and Windows
SecureLogin version:	All
Type:	Action
Usage:	RestrictVariable *variable-name password-policy*
Arguments:	
variable-name	The name of the variable to restrict.
password-policy	The name of the policy to enforce on the variable.
Description:	Monitors a *variable-name* and enforces a specified *password policy* onto the *variable-name*.

Example:	`Dialog`
	`  Title "Login"`
	`  Ctrl #1`
	`EndDialog`
	`RestrictVariable $Password PasswordPolicy2`
	`ChangePassword $Password random`

Run

Item	Description
Use with:	Startup scripts, Terminal Launcher, Web, and Windows
SecureLogin version:	All
Type:	Action
Usage:	Run *command* [*arg1]* [*arg2*] [...]
Arguments:	
command	The full path of the command to be executed.
arg1, *arg2*	An optional list of arguments to the command.
Description:	Launches the program specified in the *command* variable with the specified optional [*arg1*] [*arg2*] [...] arguments. The script doesn't wait for the launched program to complete.

Item	Description

Example:	`Dialog`
	` Title "Login"`
	` Ctrl #1`
	`EndDialog`
	`MessageBox -yesno ?result "Would you like to also run Notepad?"`
	`If ?result eq Yes`
	`Run C:\Windows\Notepad.exe`
	`EndIf`

SendKey

Item	Description
Use with:	Terminal Launcher
SecureLogin version:	2.5 and later
Type:	Action
Usage:	SendKey *text*
Arguments:	
text	The text to be typed into the emulator screen.
Description:	The SendKey command works only with Generic and Advanced Generic emulators. Use SendKey in the same manner as the Type command. Generally, the Type command is the preferred command to use. The Type command places the text into the Clipboard and then pastes it into the emulator screen. The SendKey command enters the text directly into the emulator screen. Variables do not work with the SendKey command. If you want to use variables, use the Type command.

Example:	`SendKey Username`
	`SendKey \n`
	`SendKey Password`
	`SendKey \n`

Set

Item	Description
Use with:	Startup scripts, Terminal Launcher, Web, and Windows
SecureLogin version:	All
Type:	Action
Usage:	Set *variable data*

Item	Description
Arguments:	
variable	The variable that the data is being assigned to.
data	The text or variable being read from and assigned to the variable.
Description:	Copies the value of *data* into *variable*. The *data* can be any text or another variable. However, the *variable* argument must be either ?variable or $variable.

Example:	`Dialog`
	` Title "Login"`
	` Ctrl #1`
	`EndDialog`
	`Set ?RunBefore Yes`
	`Type $Username #1`
	`Type $Password #2`
	`Click #1`

SetCursor

Item	Description
Use with:	Terminal Launcher (Only available in HLLAPI and some DDE emulators)
SecureLogin version:	All
Type:	Action
Usage 1:	SetCursor *screen-position*
Usage 2:	SetCursor *X coordinate* Y *coordinate*
Arguments:	
screen-position	On the screen, the position that the cursor should be moved to.
X coordinate	The horizontal coordinate. When *X-coordinate* is specified, a row/column conversion is carried out before the cursor is set to the position.
Y coordinate	The vertical coordinate. When *Y-coordinate* is specified, a row/column conversion is carried out before the cursor is set to the position.
Description:	Sets the cursor to a specified *screen position*. The position will be noted by a number greater than 0 (for example, SetCursor 200). If the screen position is invalid, Terminal Launcher displays an error message.
Syntax examples:	SetCursor 200 SetCursor 100 500

Example:	`SetCursor 200`
	`Type $Username`
	`Type @E`

SetFocus

Item	Description
Use with:	Windows
SecureLogin version:	All
Type:	Action
Usage:	SetFocus *#Ctrl-ID*
Arguments:	
#Ctrl-ID	The ID number of the control that the keyboard focus will be directed to.
Description:	Gives the keyboard focus to a specified control ID. For the SetFocus command to function correctly, the *#Ctrl-ID* must be valid.

Example:	`Dialog`
	` Title "Login"`
	` Ctrl #1`
	`EndDialog`
	`SetFocus #301`
	`Type $Username`
	`Type \T`
	`Type $Password`
	`Type \T`
	`Type \N`

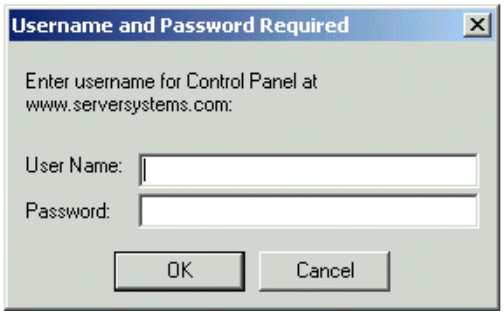
SetPlat

Item	Description
Use with:	Startup scripts, Terminal Launcher, Web, and Windows
SecureLogin version:	All
Type:	Action
Usage 1:	SetPlat *application-name*
Usage 2:	SetPlat *regex #Ctrl-ID*
Arguments:	
application-name	The application name that the variables are read from.
regex	A regular expression to be used as the application name.
#Ctrl-ID	The control ID of the regular expression to be used.

Item	Description
Description:	Forces the current script to read and write to variables that are stored under another application. If an application has a password in common with another application, the user doesn't need to enter the password twice. Instead, you can set the SetPlat *application-name* argument to switch the script engine so that it reads variables to the second application. If *application-name* doesn't exist, it will be created. SetPlat can also take a #Ctrl-ID to read from and support regular expressions.

SetPlat Example 1:

The following figure illustrates a standard dialog box for accessing a password-protected site using Netscape Navigator.



The following script illustrates a simple way to log in to this site:

```
Type $username #214
Type $password #330
Click #1
```

However, this script limits users to one username/password combination for all http-authentication Web sites accessed through Netscape. Because the dialog box illustrated above always contains the name of the Web site to log in to, such a limitation poses a significant problem.

The solution to this problem is to use a dialog block with a SetPlat statement similar to the following:

```
Dialog
  Title "Username and Password Required"
  Ctrl #330
  Ctrl #214
  Ctrl #331
  Ctrl #1
  Ctrl #2
  Setplat #331 "Enter username for .* at (.*):"
Enddialog
Type $username #214
Type $password #330
Click #1
```

The power of this script is in the following line:

```
Setplat #331 "Enter username for .* at (.*):"
```

The script first reads the following line from dialog control ID 331:

```
Enter username for Control Panel at www.movell.com:
```

The script then applies the regular expression to this text.

Regular expressions are a powerful way to manipulate text strings. However, for most purposes you can use the basic commands listed in the following table:

Basic Command	Action
* (an asterisk)	Matches any character
. (a period)	Matches zero or more of the preceding character
( ) (parentheses)	Makes the contents of the parentheses a sub-expression

After the user has run the script, the user sees the username and password saved as www.serversystems.com.

The text matched inside the parentheses then becomes the symbol application. If a dialog *#Ctrl-ID* is not specified, the symbol application will be unconditionally changed to the application specified in the *regular-expression* argument. An unconditional SetPlat command is only valid if specified before Dialog/EndDialog statements.

SetPlat Example 2:

```
Dialog
 Title "Login"
 Ctrl #1
EndDialog
ReadText #301 ?Text
Regsplit "Please enter the password for (.*) account"
?Text ?User
SetPlat ?User
Type $Username #1
Type $Password #2
Click #1
```

SetPrompt

Item	Description
Use with:	Startup scripts, Terminal Launcher, Web, and Windows
SecureLogin version:	All
Type:	Action
Usage:	SetPrompt *prompt-text*
Arguments:	
prompt-text	The customized text prompt to be displayed in the Enter SecureLogin Variables dialog box.

Item	Description
Description:	Customizes the text in the Enter SecureLogin Variables dialog boxes that are used to prompt the user for new variables. You can also use the DisplayVariables command to customize the prompt text in the dialog box.

Example 1:	The following line illustrates the default text prompt in the Enter SecureLogin Variables dialog box:
	`Please enter the following Single Sign-on variables`
	To replace this default text prompt, place the SetPrompt command at the bottom of the script. The following lines illustrate this position:
	`type $Username`
	`type $Password Password`
	`setprompt "Enter your username and password"`

Example 2:	To replace the text prompt next to any variable entry field in the Enter SecureLogin Variables dialog box, place the SetPrompt command immediately before the variable in the script.
	`Setprompt "Enter your username"`
	`Type $Username`
	`Setprompt "Enter your password"`
	`Type $Password password`
	`Setprompt "Enter your username and password"`

Strcat

Item	Description
Use with:	Startup scripts, Terminal Launcher, Web, and Windows
SecureLogin version:	All
Type:	Action
Usage:	Strcat *variable input-string1 input-string2*
Arguments:	
variable	The regular expression.
input-string1	The first data string or variable.
input-string2	The second data string or variable.
Description:	Appends the second data string to the first data string. For example, you include the following line in a script: STRCAT ?result "SecureRemote” "$username” When $username is Tim, the ?result variable contains the value "SecureRemote Tim.”

Item	Description

Example:	`Dialog`
	` Title "Login"`
	` Ctrl #1`
	`EndDialog`
	`Readtext #301 ?Username`
	`StrCat ?LoginID ?Username $Password`
	`Type ?LoginID #2`
	`Click #1`

StrLength

Item	Description
Use with:	Startup scripts, Terminal Launcher, Web, and Windows
SecureLogin version:	3.0.4
Type:	Variable Manipulator
Usage:	StrLength [*source] destination*
Arguments:	
[*source*]	The input variable. If not specified, SecureLogin reads the *destination* variable, makes the calculations, and writes over it.
destination	The output variable. Also the input variable if no source is specified.
Description:	Counts the number of characters in a variable and outputs that value to the *destination* variable. If only a *destination* variable is specified, the string is read from the destination, then the value is stored back in to it. If a *source* variable is specified, the string is read from the *source*, and the calculated value is stored in the *destination* variable. In this case, the *source* variable remains unchanged.

Example:	`Dialog`
	`Title "Login"`
	`Ctrl #1`
	`EndDialog`
	`Readtext #301 ?Password`
	`StrLength ?Password ?Length`
	`If ?Length lt 4`
	` Messagebox "Password is too short"`
	`EndIf`

StrLower

Item	Description
Use with:	Startup scripts, Terminal Launcher, Web, and Windows
SecureLogin version:	3.0.4

Item	Description
Type:	Variable Manipulator
Usage:	StrLower *destination* [*source*]
Arguments:	
destination	The output variable. Also the input variable if no source is specified.
[*source*]	The input variable. If not specified, SecureLogin reads the *destination* variable, makes the necessary changes, and writes over it.
Description:	Modifies a variable so that all the characters are lowercase. If only a *destination* variable is specified, the string is read from the destination, then stored back to it. If a *source* variable is specified, the string is read from the source, and the modified value is stored in the *destination* variable. In this case, the *source* variable remains unchanged.

Example:	`Dialog`
	`  Title "Login"`
	`  Ctrl #1`
	`EndDialog`
	`Readtext #301 ?Username`
	`StrLower ?LowerCaseUsername ?Username`
	`Type ?LowerCaseUsername #2`
	`Click #1`

StrUpper

Item	Description
Use with:	Startup scripts, Terminal Launcher, Web, and Windows
SecureLogin version:	3.0.4
Type:	Variable Manipulator
Usage:	StrUpper *variable* [*source*]
Arguments:	
destination	The output variable. Also the input variable if no source is specified.
[*source*]	The input variable. If not specified, SecureLogin reads the *destination* variable, makes the necessary changes, and writes over it.
Description:	Modifies a variable so that all the characters are uppercase. If only a *destination* variable is specified, the string is read from the destination, then stored back to it. If a *source* variable is specified, the string is read from the source, and the modified value is stored in the *destination* variable. In this case, the *source* variable remains unchanged.

Item	Description

Example:	`Dialog`
	` Title "Login"`
	` Ctrl #1`
	`EndDialog`
	`Readtext #301 ?Username`
	`StrLower ?UpperCaseUsername ?Username`
	`Type ?UpperCaseUsername #2`
	`Click #1`

Sub / Endsub

Item	Description
Use with:	Startup scripts, Terminal Launcher, Web, and Windows
SecureLogin version:	2.5 and later
Type:	Flow Control
Usage:	Sub *name* EndSub
Arguments:	
name	Any name entered to identify the subroutine.
Description:	To denote a subroutine, use the Sub/EndSub command around a block of lines within a script. You can call a subroutine by using the Call command.

Example:	`Dialog`
	` Title "Login"`
	` Ctrl #1`
	`EndDialog`
	`If -Text "Login"`
	` Call Login`
	`EndIf`
	`If -Text "Wrong Password"`
	` Call WrongPassword`
	`EndIf`
	`Sub Login`
	` Type $Username #1`
	` Type $Password #2`
	` Click #1`
	`EndSub`
	`Sub WrongPassword`
	` DisplayVariables "Enter correct password" $Password`
	`EndSub`

Submit

Item	Description
Use with:	Web components (Internet Explorer only)
SecureLogin version:	3
Type:	Flow Control
Usage:	Submit
Arguments:	None
Description:	Only use the Submit command in Web scripts and only with Internet Explorer, to allow for more control in how and when a form is submitted. When used with Netscape, the Submit command is ignored. When used with Internet Explorer, the Submit command performs a submit on the form where the first password field is found. Submit occurs automatically with most scripts. For example, the following script for Hotmail automatically does a Submit at the end: type $Username type $Password Password Submits don't automatically occur if any of the following commands are in the script: ◆ Type \n ◆ Type \t ◆ Submit ◆ Click

Example:	`Type $Username`
	`Type $Password Password`
	`Submit`

Subtract

Item	Description
Use with:	Startup scripts, Terminal Launcher, Web, and Windows
SecureLogin version:	3
Type:	Variable Manipulator
Usage:	Subtract *variable1 variable2* [?result]

Item	Description
Arguments:	
variable1	The minuend. The first argument. The number that will be subtracted from the first argument. This argument will also contain the result if the optional [?result] argument is not passed in. If you use the *variable1* argument without the [?result] argument, *variable1* must be a SecureLogin variable (either ?variable1 or $variable1). Otherwise, *variable1* can be any numeric value.
variable2	The subtrahend. The second argument. The number that will be subtracted from the first argument. *Variable2* can be a SecureLogin variable or a numeric value.
[?result]	Optional. The difference or result of the equation. See “Add” on page 14, “Divide” on page 23, and “Multiply” on page 32.

Example:	`subtract 5 2 ?res`
	`#?res will now be 3`
	`set ?res2 1`
	`subtract ?res ?res2`
	`#?res will now be 2`

Title

Item	Description
Use with:	Windows
SecureLogin version:	All
Type:	Dialog Specifier
Usage:	Title *window-title*
Arguments:	
window-title	The text to test against the window title.
Description:	Retrieves the title of the window and compares it against the string specified in the *window-title* argument. For this section of the script to run, the window title and the *window-title* argument must agree. You can use Window Finder to locate the window title. See Finding Dialog Control IDs.

Example:	`Dialog`
	` Title "Login"`
	` Ctrl #1`
	`EndDialog`
	`Type $Username #1`
	`Type $Password #2`
	`Click #1`

Type

Item	Description
Use with:	Terminal Launcher, Web, and Windows
SecureLogin version:	All
Type:	Action
Usage:	Type [-raw] *text* [*#Ctrl-ID*]
Arguments:	
[-raw]	Can only be used when you don't use a *#Ctrl-ID* argument.
text	The text to type into this area.
[*#Ctrl-ID*]	The ID number of the control that the text will be typed in.
Description:	Used to enter data (for example, usernames and passwords) into windows. You can specify a *#Ctrl-ID* as an argument after the *text* argument that will be entered. If you use the *#Ctrl-ID* argument, the *text* is sent directly to the window. As a further aid to entering *text* into a specific field or *#Ctrl-ID*, SecureLogin provides custom keys for Tab and Enter. If you don't specify the *#Ctrl-ID*, SecureLogin sends keystrokes to whichever control has focus. Only use the -raw flag when you don't use the *#Ctrl-ID* argument. By default, when SecureLogin types into a window, SecureLogin verifies that the window exists before continuing. This verification process is disabled when the -raw flag is set, causing SecureLogin to log in to whichever window has focus. This is useful for typing after a Delay that is a step in the login process.

Example:	`Dialog`
	`  Title "Login"`
	`  Ctrl #1`
	`EndDialog`
	`Type $Username #1`
	`Type $Password #2`
	`Click #1`

Type Commands Used with Windows

SecureLogin can send special keyboard keystrokes to Windows and Internet-based applications to emulate the user's keyboard entry. These special commands include the ability to select Menu items, send ALT, and send other keyboard combinations.

The following table illustrates keyboard sequences that you can use to select a menu item within an application:

Desired Result	Sequence
Select a file	Type \Alt+F

Desired Result	Sequence
Select a tool	Type T
Select Change Password	Type C
Select a given option	Type \Alt+*x* Type \Ctr+*x* Type \Shift+*x*
Send the Backspace key	Type \b
Send the Delete key	Type \d
Send the End key	Type \e
Send the Home key	Type \h
Send the Enter key	Type \N Type \n
Send the Tab key	Type \T
Send the Shift-Tab keys	Type \-T
Send the value of the password variable to the control ID #1056	Type $Password #1056

The following commands pass keystrokes through the script to the window that the script is working in.

To Send This Result	Type This Sequence
The Spacebar	Type |32
The End key	Type |35
The Home key	Type |36
The Left-arrow key	Type |37
The Up-arrow key	Type |38
The Right-arrow key	Type |39
The Down-arrow key	Type |40

Type Commands Used with the Web

The following table provides information about the Type command used with the Web.

To Send This Result	Type This Sequence
The Enter key	Type \N Type \n

To Send This Result	Type This Sequence
The Tab key	Type \T Type \t
Send the value of the password variable to the control ID #1056	Type $Password #1056
Hidden passwords	Type $Password Password

NOTE: Always add *password* to the end of any $Password variable lines in the script. The SecureLogin method of scripting requires that a password field is defined by using the *password* flag.

Type Commands Used with Terminal Launcher

Terminal Launcher uses the High Level Language Application Programming Interface (HLLAPI) to interface with a wide range of mainframe emulators which implement this programming standard.

The following table lists the @ commands that you can use in the SecureLogin script Type command. These commands perform specific emulator and mainframe functions. For example, you can send an Enter, Tab, or cursor key or issue a mainframe emulator print screen or reset function.

The @ commands are used in script language in the following format:

- TYPE @ *command*
- WAITFORTEXT "Login:”
- Type $username
- Type @T
- Type $password
- Type @E

The Type Command	Meaning	The Type Command	Meaning
@B	Left Tab	@A@C	Test
@C	Clear	@A@D	Word Delete
@D	Delete	@A@E	Field Exit
@E	Enter	@A@F	Erase Input
@F	Erase EOF	@A@H	System Request
@H	Help	@A@I	Insert Toggle
@I	Insert	@A@J	Cursor Select
@J	Jump (Set Focus)	@A@L	Cursor Left Fast
@L	Cursor Left	@A@Q	Attention
@N	New Line	@A@R	Device Cancel (Cancels Print Presentation Space)

The Type Command	Meaning	The Type Command	Meaning
@O	Space	@A@T	Print Presentation Space
@P	Print	@A@U	Cursor Up Fast
@R	Reset	@A@V	Cursor Down Fast
@T	Right Tab	@A@Z	Cursor Right Fast
@U	Cursor Up	@A@9	Reverse Video
@V	Cursor Down	@A@b	Underscore
@X*	DBCS (Reserved)	@A@c	Reset Reverse Video
@Y	Caps Lock (No action)	@A@d	Red
@Z	Cursor Right	@A@e	Pink
@0	Home	@A@f	Green
@1	PF1/F1	@A@g	Yellow
@2	PF2/F2	@A@h	Blue
@3	PF3/F3	@A@i	Turquoise
@4	PF4/F4	@A@l	Reset Host Colors
@5	PF5/F5	@A@j	White
@6	PF6/F6	@A@t	Print (Personal Computer)
@7	PF7/F7	@A@y	Forward Word Tab
@8	PF8/F8	@A@z	Backward Word Tab
@9	PF9/F9	@A@-	Field -
@a	PF10/F10	@A@<	Record Backspace
@b	PF11/F11	@A@+	Field +
@c	PF12/F12	@S@x	Dup
@d	PF13	@S@E	Print Presentation Space or Host
@e	PF14	@S@y	Field Mark
@f	PF15	@X@c	Split Vertical Bar (¦)
@g	PF16	@X@7	Forward Character
@h	PF17	@X@6	Display Attribute
@i	PF18	@X@5	Generate SO/SI
@j	PF19	@X@1	Display SO/SI
@k	PF20	@M@0	VT Numeric Pad 0

The Type Command	Meaning	The Type Command	Meaning
@l	PF21	@M@1	VT Numeric Pad 1
@m	PF22	@M@2	VT Numeric Pad 2
@n	PF23	@m@3	VT Numeric Pad 3
@o	PF24	@M@4	VT Numeric Pad 4
@q	End	@M@5	VT Numeric Pad 5
@s	ScrLk (No action)	@M@6	VT Numeric Pad 6
@t	Num Lock (No action)	@M@7	VT Numeric Pad 7
@u	Page Up	@M@8	VT Numeric Pad 8
@v	Page Down	@M@9	VT Numeric Pad 9
@x	PA1	@M@-	VT Numeric Pad
@y	PA2	@M@,	VT Numeric Pad
@z	PA3	@M@.	VT Numeric Pad
@M@h	VT Hold Screen	@M@e	VT Numeric Pad Enter
@M@N	Control Code SO	@M@f	VT Edit Find
@M@M	Control Code CR	@M@i	VT Edit Insert
@M@L	Control Code FF	@M@r	VT Edit Remove
@M@K	Control Code VT	@M@s	VT Edit Select
@M@J	Control Code LF	@M@p	VT Edit Previous Screen
@M@I	Control Code HT	@M@n	VT Edit Next Screen
@M@H	Control Code BS	@M@a	VT PF1
@M@G	Control Code BEL	@M@b	VT PF2
@M@F	Control Code ACK	@M@c	VT PF3
@M@(space)	Control Code NUL	@M@d	VT PF4
@M@E	Control Code ENQ	@M@O	ControlCode S1
@M@D	Control Code EOT	@M@Q	ControlCode DC1
@M@C	Control Code ETX	@M@P	ControlCode DLE
@M@B	Control Code STX	@M@A	ControlCode SOH

WaitForFocus

Item	Description
Use with:	Windows
SecureLogin version:	All
Type:	Flow Control
Usage:	WaitForFocus #*Ctrl-ID repeat-loops*
Arguments:	
#Ctrl-ID	The ID number of the control that will have the focus.
repeat-loops	Optional. The number of repeat loops that will run.
Description:	Suspends the running of the script until the *#Ctrl-ID* has received keyboard focus or until the *repeat loops* argument expires. The *repeat-loops* argument is an optional value, which defines the number of loop cycles that will be run. The *repeat-loops* value defaults to 3000 loops if nothing is set. As soon as focus is received, the script continues. As the following line illustrates, you can set the *repeat-loop* to never expire by setting the value to a negative number: `WaitForFocus "#1065" "-1"` If the #Ctrl-ID is set to 0 (zero), it will loop until the window defined in the Dialog / EndDialog statement is given keyboard focus. **NOTE:** Do not place WaitForFocus commands within Dialog / EndDialog statements.
Syntax examples:	WaitForFocus #301 WaitForFocus #301 2000 WaitForFocus #301 0 WaitForFocus #301 -1

Example:	`Dialog`
	`  Title "Login"`
	`  Ctrl #1`
	`EndDialog`
	`WaitForFocus #301 -1`
	`Type $Username`
	`Type \T`
	`Type $Password`
	`Type \N`

WaitForText

Item	Description
Use with:	Terminal Launcher
SecureLogin version:	All

Item	Description
Type:	Flow Control
Usage:	WaitForText *text*
Arguments:	
text	The text that the script is waiting for.
Description:	Causes Terminal Launcher to wait for the specified *text* to be displayed before continuing. This command is essential because the user will always want to wait for particular text to be displayed on the screen before continuing. Otherwise, the user might enter *text* too soon. The *text* can appear anywhere on the terminal screen. The *text* is case-sensitive. If the *text* is written in the wrong case, Terminal Launcher pauses and tries to find the correct *text* in the correct case, pausing until the terminal screen times out. If WaitForText is not working, try leaving the initial letter off the *text*, so that you avoid any conflict with case. For example, the following line will work regardless of whether "login" is presented on the terminal screen as “Login” or “login”. `WaitForText "ogin″` However, WaitForText “Login” will only work if “login” is presented on the screen as “Login”.

Example:	`WaitForText "Login:"`
	`Type $Username`
	`Type @E`
	`WaitForText "Password:"`
	`Type $Password`
	`Type @E`

A Quick-Reference Chart

This section provides a quick-reference chart of commands used in SecureLogin scripts. The chart lists the following:

- The platform that the command is used with (Startup scripts, Terminal Launcher, Web, or Windows)
- The type of command (Action, Dialog Specifier, Flow Control, or Variable Manipulator)
- The SecureLogin version that the command is used with (All, 2.5 and later, 3, or 3.0.4)

Command	Use With	Type	SecureLogin Version
AAVerify	SS, TL, Web, Win	Action	3.0.4
Add	SS, TL, Web, Win	Variable Manipulator	3
BeginSplashScreen	TL	Action	3.0.4
Break	SS, TL, Web, Win	Action	2.5
Call	SS, TL, Web, Win	Flow Control	2.5
ChangePassword	SS, TL, Web, Win	Action	All
Class	Win	Dialog Specifier	All
Click	Web, Win	Action	All
ConvertTime	SS, TL, Web, Win	Variable Manipulator	3.0.4
Ctrl	Win	Dialog Specifier	All
Delay	SS, TL, Web, Win	Action	All
Dialog/EndDialog	Win	Dialog Specifier	All
DisplayVariables	SS, TL, Web, Win	Action	All
Divide	SS, TL, Web, Win	Variable Manipulator	3
EndScript	SS, TL, Web, Win	Action	All
GetCommandLine	Win	Action	3.0.4
GetText	Web	Action	2.5
GetURL	Web	Action	2.5
GoToURL	Web	Action	2.5

Command	Use With	Type	SecureLogin Version
If/Else/EndIF	SS, TL, Web, Win	Flow Control	All
Increment/Decrement	SS, TL, Web, Win	Flow Control	All
KillApp	SS, TL, Web, Win	Flow Control	All
Local	SS, TL, Web, Win	Variable Manipulator	3
MessageBox	SS, TL, Web, Win	Action	All
Multiply	SS, TL, Web, Win	Variable Manipulator	3
OnException	SS, TL, Web, Win	Flow Control	3.0.4
Parent/EndParent	Win	Flow Control	All
PickListAdd	SS, TL, Web, Win	Action	All
PickListDisplay	SS, TL, Web, Win	Action	All
ReadText	TL, Win	Action	All
RegSplit	SS, TL, Web, Win	Action	All
Repeat/EndRepeat	SS, TL, Web, Win	Action	All
RestrictVariable	SS, TL, Web, Win	Action	All
Run	SS, TL, Web, Win	Action	All
SendKey	TL	Action	2.5
Set	SS, TL, Web, Win	Action	All
SetCursor	TL	Action	All
SetFocus	Win	Action	All
SetPlat	SS, TL, Web, Win	Action	All
SetPrompt	SS, TL, Web, Win	Action	All
StrCat	SS, TL, Web, Win	Action	All
StrLength	SS, TL, Web, Win	Variable Manipulator	3.0.4
StrLower	SS, TL, Web, Win	Variable Manipulator	3.0.4
StrUpper	SS, TL, Web, Win	Variable Manipulator	3.0.4
Sub/EndSub	SS, TL, Web, Win	Flow Control	2.5
Submit	Web	Flow Control	3
Subtract	SS, TL, Web, Win	Variable Manipulator	3
Title	Win	Dialog Specifier	All
Type	TL, Web, Win	Action	All

Command	Use With	Type	SecureLogin Version
WaitForFocus	Win	Flow Control	All
WaitForText	TL	Flow Control	All

B What's New or Updated

This section contains new or updated information on script commands. The information is new since SecureLogin 3.0.3.

This documentation is also provided on the Web in two formats: HTML and PDF. The HTML and PDF documentation are both kept up-to-date with the documentation changes listed in this section. See SecureLogin 3.0 (http://www.novell.com/documentation/lg/securelogin30/index.html).

If you need to know whether a copy of the PDF documentation you are using is the most recent, check the date that the PDF file was published. The date is in the Legal Notices section, which immediately follows the title page.

New or updated documentation was published on the following dates:

- "November 21, 2002" on page 63
- "December 10, 2002" on page 64

November 21, 2002

What's New

A New Guide

Script Commands is a new guide in the Novell® SecureLogin 3.0.4 set of documents. The SecureLogin Commands chapter was removed from the *SecureLogin Administration Guide* and placed in *Script Commands*.

Also, content relating to script commands was removed from the Administering Scripts chapter and placed in *Script Commands*.

New Commands

The following commands are new to SecureLogin 3.0.4:

- "BeginSplashScreen / EndSplashScreen" on page 14
- "ConvertTime" on page 19
- "GetCommandline" on page 24
- "OnException/ClearException" on page 32
- "StrLength" on page 46
- "StrLower" on page 46
- "StrUpper" on page 47

Quick-Reference Chart

Script Commands contains a quick-reference chart on the commands. See Appendix A, “Quick-Reference Chart,” on page 59.

What’s Updated

The AAVerify command was updated to include information on arguments.

Most example scripts for commands were updated.

December 10, 2002

The Include command was added. See “Include” on page 27.

Information on Usage for the StrLength command was corrected. See “StrLength” on page 46.

The Subtract entry in the Quick-Reference chart was corrected to show SecureLogin support for all "use with" platforms. See Appendix A, “Quick-Reference Chart,” on page 59.