

Application Definition Guide

Novell® SecureLogin

6.1 SP1

June, 2009

www.novell.com



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.

Any products or technical information provided under this Agreement may be subject to U.S. export controls and the trade laws of other countries. You agree to comply with all export control regulations and to obtain any required licenses or classification to export, re-export or import deliverables. You agree not to export or re-export to entities on the current U.S. export exclusion lists or to any embargoed or terrorist countries as specified in the U.S. export laws. You agree to not use deliverables for prohibited nuclear, missile, or chemical biological weaponry end uses. See the [Novell International Trade Services Web page \(http://www.novell.com/info/exports/\)](http://www.novell.com/info/exports/) for more information on exporting Novell software. Novell assumes no responsibility for your failure to obtain any necessary export approvals.

Copyright © 2004-2009 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., has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed on the [Novell Legal Patents Web page \(http://www.novell.com/company/legal/patents/\)](http://www.novell.com/company/legal/patents/) and one or more additional patents or pending patent applications in the U.S. and in other countries.

Novell, Inc.
404 Wyman Street, Suite 500
Waltham, MA 02451
U.S.A.
www.novell.com

Online Documentation: To access the latest online documentation for this and other Novell products, see the [Novell Documentation Web page \(http://www.novell.com/documentation\)](http://www.novell.com/documentation).

Novell Trademarks

For Novell trademarks, see the [Novell Trademark and Service Mark list \(http://www.novell.com/company/legal/trademarks/tmlist.html\)](http://www.novell.com/company/legal/trademarks/tmlist.html).

Third-Party Materials

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

Contents

About This Guide	9
1 Quick Command Reference	11
2 Application Definition Language: an Overview	21
2.1 Using Application Definitions	21
2.2 Advantages of Using Application Definitions	21
2.3 Defining Applications Enabled for Single Sign-On	22
2.4 Corporate Definitions	22
2.5 What Is an Application Definition?	22
2.5.1 Using Dialog Specifier Commands	23
2.5.2 Reading from and Writing to Variables	23
3 Managing Application Definitions	25
3.1 Application Definition Checklist	25
3.2 Exporting and Importing Predefined Applications and Application Definitions	25
3.2.1 Exporting Individual Applications	26
3.2.2 Importing Individual Applications	29
3.3 Modifying Predefined Applications and Application Definitions	30
3.3.1 Building an Application Definition in the Personal Management Utility	32
3.4 Windows Application Definition Tools	36
3.4.1 Finding Application Details with Window Finder	36
3.4.2 Finding Application Details with the Login Watcher	38
3.5 Application Definition Elements	40
4 Application Definition Variables	43
4.1 Types of Variables	43
4.1.1 Using a Variable to Change the Default Platform	43
4.1.2 Directory Attribute Variables	43
4.1.3 Stored Variables	44
4.1.4 Runtime Variables	45
4.1.5 Passticket Variables	45
4.1.6 SecureLogin Supported Variables	46
4.2 Application Definition Variables	47
4.2.1 Symbols Used	48
4.2.2 Capitalization	48
4.2.3 Comments	48
4.2.4 Switches	48
4.2.5 Variables	49
4.2.6 Indent Sections	49
4.2.7 Blank Line Between Sections	49
4.2.8 Writing Subroutine Sections	50
4.2.9 Quotation Marks	50
4.2.10 Password Policy Names	50
4.2.11 Regular Expressions	51

5 Command Reference 53

5.1	Command Reference Conventions	53
5.1.1	Command Information	53
5.1.2	Web Wizard Application Definition Conventions	54
5.1.3	Integrating Novell Audit	55
5.1.4	One-Time Passwords	56
5.2	Commands	56
5.2.1	AAVerify	59
5.2.2	ADD	62
5.2.3	Attribute	63
5.2.4	AuditEvent	64
5.2.5	BeginSplashScreen/EndSplashScreen	64
5.2.6	BooleanInput	65
5.2.7	Break	66
5.2.8	Call	68
5.2.9	ChangePassword	69
5.2.10	Class	70
5.2.11	ClearPlat	71
5.2.12	ClearSite	74
5.2.13	Click	74
5.2.14	ConvertTime	77
5.2.15	Ctrl	77
5.2.16	DebugPrint	78
5.2.17	Decrement	79
5.2.18	Delay	80
5.2.19	Dialog/EndDialog	81
5.2.20	DisplayVariables	82
5.2.21	Divide	83
5.2.22	DumpPage	84
5.2.23	EndScript	85
5.2.24	Event/Event Specifiers	85
5.2.25	FocusInput	86
5.2.26	GenerateOTP	87
5.2.27	GetCheckBoxState	89
5.2.28	GetCommandLine	90
5.2.29	GetEnv	91
5.2.30	GetHandle	91
5.2.31	GetIni	92
5.2.32	GetMD5	92
5.2.33	GetReg	93
5.2.34	GetSessionName	94
5.2.35	GetText	94
5.2.36	GetURL	95
5.2.37	GoToURL	96
5.2.38	Highlight	96
5.2.39	If/Else/EndIf	97
5.2.40	Include	100
5.2.41	Increment	101
5.2.42	KillApp	102
5.2.43	Local	103
5.2.44	MatchDomain	104
5.2.45	MatchField	105
5.2.46	MatchForm	107
5.2.47	MatchOption	109
5.2.48	MatchReferer	110
5.2.49	MatchTitle	111
5.2.50	MatchURL	112
5.2.51	MessageBox	113

5.2.52	Multiply	115
5.2.53	OnException/ClearException	116
5.2.54	Parent/EndParent	121
5.2.55	PickListAdd	123
5.2.56	PickListDisplay	125
5.2.57	PositionCharacter	126
5.2.58	PressInput	127
5.2.59	ReadText	128
5.2.60	RegSplit	131
5.2.61	ReLoadPlat	132
5.2.62	Repeat/EndRepeat	134
5.2.63	RestrictVariable	135
5.2.64	Run	138
5.2.65	Select	139
5.2.66	SelectListBoxItem	140
5.2.67	SelectOption	140
5.2.68	SendKey	141
5.2.69	Set	142
5.2.70	SetCheckBox	143
5.2.71	SetCursor	144
5.2.72	SetFocus	145
5.2.73	SetPlat	146
5.2.74	SetPrompt	149
5.2.75	-SiteDeparted	150
5.2.76	Site/Endsite	151
5.2.77	StrCat	153
5.2.78	StrLength	154
5.2.79	StrLower	155
5.2.80	StrUpper	155
5.2.81	Sub/EndSub	156
5.2.82	Submit	157
5.2.83	Subtract	158
5.2.84	Tag/EndTag	160
5.2.85	TextInput	160
5.2.86	Title	161
5.2.87	Type	162
5.2.88	Using the Type Command to Send Keyboard Commands	166
5.2.89	WaitForFocus	167
5.2.90	WaitForText	169
6 Testing Application Definitions		171
6.1	Using the SecureLogin Test Application	171
6.1.1	Example Application Definition for the Test Application	171
6.1.2	About the Application Definition	172
6.1.3	Dialog Boxes	173
7 Reference Commands and Keys		177
7.1	Typing Keys	177
7.2	Windows Keyboard Functions	177
7.3	Terminal Emulator Commands	182
8 Application Definition Commands for SNMP Alerts		185
8.1	Creating an SNMP Alert	185

About This Guide

This guide provides you with information that enables you to build application definitions or modify predefined application definitions.

This document contains information on the following:

- ♦ Chapter 1, “Quick Command Reference,” on page 11
- ♦ Chapter 2, “Application Definition Language: an Overview,” on page 21
- ♦ Chapter 3, “Managing Application Definitions,” on page 25
- ♦ Chapter 4, “Application Definition Variables,” on page 43
- ♦ Chapter 5, “Command Reference,” on page 53
- ♦ Chapter 6, “Testing Application Definitions,” on page 171
- ♦ Chapter 7, “Reference Commands and Keys,” on page 177
- ♦ Chapter 8, “Application Definition Commands for SNMP Alerts,” on page 185

Document Scope

The instructions and example provided in this guide apply to Microsoft* Windows* 2000 and 2003 Active Directory* environments with a directory server managed through an administration workstation.

If you have implemented alternate versions of the operating system, read the relevant documentation or contact Novell Support for help.

Audience

This guide is intended for:

- ♦ Network Administrators
- ♦ System Administrators
- ♦ IT Support Staff

Feedback

We want to hear your comments and suggestions about this manual and the other documentation included with this product. Please use the User Comments feature at the bottom of each page of the online documentation, or go to the [Novell Documentation Feedback \(http://www.novell.com/documentation/feedback.html\)](http://www.novell.com/documentation/feedback.html) and enter your comments there.

Documentation Updates

For the most recent version of the *Novell SecureLogin 6.1 SPI Application Definition Guide*, visit the [Novell Documentation Web site. \(http://www.novell.com/documentation/securelogin61/index.html\)](http://www.novell.com/documentation/securelogin61/index.html)

Additional Documentation

The *Application Definition Guide* is part of the documentation set for Novell SecureLogin 6.1 SP1 release.

Other documents part of this release are:

- ◆ *Novell SecureLogin 6.1 SP1 Installation Guide*
- ◆ *Novell SecureLogin 6.1 SP1 Administration Guide*
- ◆ *Novell SecureLogin 6.1 SP1 Citrix and Terminal Services Guide*
- ◆ *Novell SecureLogin 6.1 SP1 User Guide*
- ◆ Quick Start: “*NMAS Login Method and Login ID Snap-In for pcProx*”
- ◆ Readme: Available online at the [Novell Documentation Web site](http://www.novell.com/documentation/securelogin61sp1/index.html). (<http://www.novell.com/documentation/securelogin61sp1/index.html>)

Documentation Conventions

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

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

Quick Command Reference

1

This section provides a quick reference to the commands.

Table 1-1 Quick Command Reference

Command	Description
#	<p>Use the this symbol to define a line of text as a comment field. Comment fields are used to leave notes.</p> <p>For detailed information, see “#” on page 41.</p>
""	<p>Use quotation marks to group together text or variables containing spaces. Quotation marks are used with commands like <code>Type</code>, <code>MessageBox</code>, and <code>If - Text</code>.</p> <p>For detailed information, see “” on page 41.</p>
\$	<p>Use the dollar sign to define the use of a SecureLogin variable stored in the directory for later use by that user.</p> <p>For detailed information, see “\$” on page 41.</p>
?	<p>Use the question mark to define the use of a runtime variable.</p> <p>The values of these variables are not stored in the directory. They are reset each time SecureLogin is started.</p> <p>For detailed information, see “?” on page 41.</p>
%	<p>Use the percentage sign to define the use of a directory attribute. The attributes that are available vary depending on the directory in use, and the setup of the directory.</p> <p>For detailed information, see “%” on page 42.</p>
!	<p>Use the exclamation mark to define the use of a passticket. A passticket is a one-time password (OTP) that is generated by using a combination of an encryption key, encryption offset, and the current time.</p> <p>For detailed information, see “!” on page 42.</p>
\	<p>Use the backslash with the <code>Type</code> and <code>Send Key</code> commands to specify the use of a special function.</p> <p>For detailed information, see “\” on page 42.</p>
@	<p>Use this symbol in the same way as the backslash symbol, except its use is limited to HLLAPI enabled emulators.</p> <p>For detailed information, see “@” on page 42.</p>
-	<p>Use the hyphen as a switch within several commands, such as <code>If</code> and <code>Type</code>.</p> <p>For detailed information, see “-” on page 42.</p>

Command	Description
AAVerify	<p>Use <code>AAVerify</code> with SecureLogin Advanced Authentication or Novell NMAS to verify the user. It is typically used before the application Username and Password are retrieved and entered into the logon box.</p> <p>For detailed information, see Section 5.2.1, “AAVerify,” on page 59.</p>
ADD	<p>Adds one number to another. The numbers can be hard-coded into the application definition, or they can be variables. The result can be the output of another variable, or one of the original numbers.</p> <p>For detailed information, see Section 5.2.2, “ADD,” on page 62.</p>
Attribute	<p>Use the <code>Attribute</code> specifier in conjunction with the <code>Tag/EndTag</code> command to specify which HTML attributes and attribute values must exist for that particular HTML tag.</p> <p>For detailed information, see Section 5.2.3, “Attribute,” on page 63.</p>
AuditEvent	<p>Use the <code>AuditEvent</code> to audit the following events from an application definition:</p> <ul style="list-style-type: none"> ◆ SecureLogin client started ◆ SecureLogin client exited ◆ SecureLogin client activated by user ◆ SecureLogin client deactivated by user ◆ Password provided to an application by a script ◆ Password changed by the user in response to a <code>changepassword</code> command ◆ Password changed automatically in response to a <code>changepassword</code> command <p>For detailed information, see Section 5.2.4, “AuditEvent,” on page 64.</p>
BeginSplashScreen/ EndSplashScreen	<p>Use to display a Novell splash screen across the whole Terminal Emulator window. This is used to mask any flashing produced by SecureLogin scraping the screen for text. A <code>Delay</code> command at the start of the application definition ensures that the emulator window is in place before the splash screen is displayed.</p> <p>For detailed information, see Section 5.2.5, “BeginSplashScreen/EndSplashScreen,” on page 64.</p>
BooleanInput	<p>Use <code>BooleanInput</code> within a <code>site</code> block to set the state of a Boolean field (either a check box or radio button).</p> <p>For detailed information, see Section 5.2.6, “BooleanInput,” on page 65.</p>
Break	<p>Use <code>Break</code> within the <code>Repeat/EndRepeat</code> commands to break out of a repeat loop.</p> <p>For detailed information, see Section 5.2.7, “Break,” on page 66.</p>
Call	<p>Use the <code>Call</code> command to call and run a subroutine. When a subroutine is called, the application definition begins executing from the first line of the subroutine.</p> <p>For detailed information, see Section 5.2.8, “Call,” on page 68.</p>

Command	Description
ChangePassword	<p>Use the <code>ChangePassword</code> command to change a single variable and is used in scenarios where password expiry is an issue. Set the <code><Variable></code> to the new password.</p> <p>For detailed information, see Section 5.2.9, "ChangePassword," on page 69.</p>
Class	<p>When a window is created, it is based on a template known as a window class. The <code>Class</code> command checks to see if the class of the newly created window matches its <code><Window-Class></code> argument.</p> <p>For detailed information, see Section 5.2.10, "Class," on page 70.</p>
ClearPlat	<p>Use to reset the last chosen platform, causing subsequent calls to <code>ReLoadPlat</code> to do nothing.</p> <p>For detailed information, see Section 5.2.11, "ClearPlat," on page 71.</p>
ClearSite	<p>Use within a <code>Site</code> block to clear the 'matched' status for a given site.</p> <p>For detailed information, see Section 5.2.12, "ClearSite," on page 74.</p>
Click	<p>When used with windows applications, the <code>Click</code> command sends a click instruction to the specified <code><#Ctrl-ID></code>.</p> <p>For detailed information, see Section 5.2.13, "Click," on page 74.</p>
ConvertTime	<p>Use to convert a numeric time value, for example, <code>?CurrTime(system)</code>, into a legible format and store it in <code><String Time></code>.</p> <p>For detailed information, see Section 5.2.14, "ConvertTime," on page 77.</p>
Ctrl	<p>Use the <code>Ctrl</code> command to determine if a window contains the control expressed in the <code><#Ctrl-ID></code> argument. The control ID number is a constant that is established at the time a program is compiled.</p> <p>For detailed information, see Section 5.2.15, "Ctrl," on page 77.</p>
DebugPrint	<p>Use the <code>DebugPrint</code> command to display the text specified in the <code><Data></code> variable on a Debug console. The command can take any number of text arguments, including variables, (for example, <code>DebugPrint "The user " \$Username " has just been logged onto the system"</code>).</p> <p>For detailed information, see Section 5.2.16, "DebugPrint," on page 78.</p>
Decrement	<p>Use the <code>Decrement</code> command to subtract from a specified variable. For example, you can use <code>Decrement</code> to count the number of passes a particular application definition has made.</p> <p>For detailed information, see Section 5.2.17, "Decrement," on page 79.</p>
Delay	<p>Use the <code>Delay</code> command to delay the execution of the application definition for the time specified in the <code><Time Period></code> argument.</p> <p>For detailed information, see Section 5.2.18, "Delay," on page 80.</p>
Dialog/EndDialog	<p>Use the <code>Dialog/EndDialog</code> command to identify the beginning and end of a dialog specification block respectively. You can use these commands to construct a dialog specification block, which consists of a series of dialog specification statements (for example <code>Ctrl</code> and <code>Title</code>).</p> <p>For detailed information, see Section 5.2.19, "Dialog/EndDialog," on page 81.</p>

Command	Description
DisplayVariables	<p>Use the <code>DisplayVariables</code> command to display a dialog box that lists the user's stored variables (for example, <code>\$Username</code> and <code>\$Password</code>) for the current application.</p> <p>For detailed information, see Section 5.2.20, "DisplayVariables," on page 82.</p>
Divide	<p>Use to divide one number by another. The numbers can be hard coded into the application definition, or they can be variables. The result can be output to another variable, or to one of the original numbers.</p> <p>For detailed information, see Section 5.2.21, "Divide," on page 83.</p>
DumpPage	<p>Use the <code>DumpPage</code> command to provide information about the current Web page. Use for debugging Web page application definitions.</p> <p>For detailed information, see Section 5.2.22, "DumpPage," on page 84.</p>
EndScript	<p>Use the <code>EndScript</code> command to immediately terminate execution of the application definition.</p> <p>For detailed information, see Section 5.2.23, "EndScript," on page 85.</p>
Event/Event Specifiers	<p>Application definitions generally execute at the point when an application window is created. This corresponds to the <code>WM_CREATE</code> message that is received from an application window at startup.</p> <p>By adding the Event specifier to a dialog block, you can override this behavior, so that an application definition only executes when the specified message is generated. If no Event specifier is given, it is equivalent to Event <code>WM_CREATE</code>.</p> <p>For detailed information, see Section 5.2.24, "Event/Event Specifiers," on page 85.</p>
FocusInput	<p>Use within a <code>Site</code> Block to focus on an input field based on the Boolean value of <code>"focus"</code>.</p> <p>For detailed information, see Section 5.2.25, "FocusInput," on page 86.</p>
GenerateOTP	<p>Used to generate a one time password (OTP) as an authentication method in lieu of a traditional fixed and static password.</p> <p>The OTP is a hard token generated by the Vasco Digipass, RSA SecureID Token and Mini Token products or might be produced by a soft token generator functionality embedded in SecureLogin.</p> <p>For detailed information, see Section 5.2.26, "GenerateOTP," on page 87.</p>
GetCheckBoxState	<p>Use the <code>GetCheckBoxState</code> command to return the current state of the specified checkbooks. For detailed information, see Section 5.2.27, "GetCheckBoxState," on page 89.</p>
GetCommandLine	<p>Use the <code>GetCommandLine</code> command to capture the full command line of the program that is loaded, and save it to the specified variable.</p> <p>For detailed information, see Section 5.2.28, "GetCommandLine," on page 90.</p>
GetEnv	<p>Use the <code>GetEnv</code> command to read the value of an environment variable and save it in the specified <code><variable></code>.</p> <p>For detailed information, see Section 5.2.29, "GetEnv," on page 91.</p>

Command	Description
GetHandle	<p>Use <code>GetHandle</code> to capture the unique handle of the window on which the Windows application definition script is activated.</p> <p>For detailed information, see Section 5.2.30, "GetHandle," on page 91.</p>
GetIni	<p>Use the <code>GetIni</code> command to read data from the INI file.</p> <p>For detailed information, see Section 5.2.31, "GetIni," on page 92.</p>
GetMD5	<p>Use the <code>GetMD5</code> command to generate an MD5 hash value of the current process the script is running for. <code>GetMD5</code> works only with the Win32 scripts.</p> <p>For detailed information, see Section 5.2.32, "GetMD5," on page 92.</p>
GetReg	<p>Use the <code>GetReg</code> command to read data from the registry and save it in the specified <code><variable></code>.</p> <p>For detailed information, see Section 5.2.33, "GetReg," on page 93.</p>
GetSessionName	<p>Use the <code>GetSessionName</code> command to find the current HLLAPI session name that is used to connect and return it to the specified variable.</p> <p>For detailed information, see Section 5.2.34, "GetSessionName," on page 94.</p>
GetText	<p>Use the <code>GetText</code> command to get all of the text from the screen and save it to the specified variable. It is used in a large Web application definition that might contain several <code>If-Text</code> statements.</p> <p>For detailed information, see Section 5.2.35, "GetText," on page 94.</p>
GetURL	<p>Use the <code>GetURL</code> command to capture the URL of the site that is loaded and save it to the specified variable.</p> <p>For detailed information, see Section 5.2.36, "GetURL," on page 95.</p>
GoToURL	<p>Use the <code>GoToURL</code> command to make the browser navigate to the specified <code><URL></code>. By default the command opens the new Web page in the main window, rather than the frame that started the application definition.</p> <p>For detailed information, see Section 5.2.36, "GetURL," on page 95.</p>
If/Else/EndIf	<p>Use the <code>If</code> command to establish a block to execute if the expression supplied is true. The <code>Else</code> command works inside an <code>If</code> block. The <code>Else</code> command is executed if the operator in the <code>If</code> block is false. Use the <code>EndIf</code> command to terminate the <code>If</code> block.</p> <p>For detailed information, see Section 5.2.39, "If/Else/EndIf," on page 97.</p>
Include	<p>Use the <code>Include</code> command to share commonly used application definition commands by multiple applications. The application definition identified by <code><Platform-Name></code> is included at execution time into the calling application definition. The application definition included with the <code>Include</code> command must consist of commands supported by the calling application.</p> <p>For detailed information, see Section 5.2.40, "Include," on page 100.</p>
Increment	<p>Use the <code>Increment</code> command to add to a specified variable. For example, you can use <code>increment</code> to count the number of passes a particular application definition has made.</p> <p>For detailed information, see Section 5.2.41, "Increment," on page 101.</p>

Command	Description
KillApp	<p>Use to terminate an application.</p> <p>For detailed information, see Section 5.2.42, "KillApp," on page 102.</p>
Local	<p>Use the <code>Local</code> command to declare that a runtime variable will only exist for the lifetime of the application definition. Local runtime variables are used in the same way as normal runtime variables and are still written as <code>?Variable</code>.</p> <p>For detailed information, see Section 5.2.43, "Local," on page 103.</p>
MatchDomain	<p>Use <code>MatchDomain</code> inside a site block to filter a site based on its domain. If the domain does not match, the site block fails to match.</p> <p>For detailed information, see Section 5.2.44, "MatchDomain," on page 104.</p>
MatchField	<p>Use <code>MatchField</code> to filter a form based on the presence of a particular field. If the field fails to match and it is not specified as optional, then the parent form fails to match.</p> <p>For detailed information, see Section 5.2.45, "MatchField," on page 105.</p>
MatchForm	<p>Use <code>MatchForm</code> to filter a site based on the presence of a particular field. If the field fails to match and it is not specified as optional, then the site fails to match.</p> <p>For detailed information, see Section 5.2.46, "MatchForm," on page 107.</p>
MatchOption	<p>Use the <code>MatchOption</code> command to filter a field based on the presence of a particular option.</p> <p>For detailed information, see Section 5.2.47, "MatchOption," on page 109.</p>
MatchReferer	<p>Use <code>MatchReferer</code> inside a <code>Site/EndSite</code> block to match or filter a site based on a referrer.</p> <p>For detailed information, see Section 5.2.48, "MatchReferer," on page 110.</p>
MatchTitle	<p>Used inside a site block. <code>MatchTitle</code> is used to filter a site based on its title. If the site title does not match, the site block fails to match.</p> <p>For detailed information, see Section 5.2.49, "MatchTitle," on page 111.</p>
MatchURL	<p>Use <code>MatchURL</code> inside a site block to match or filter an HTML page within a site based on its URL. The URL can be a complex Web address or a secure Web site.</p> <p>For detailed information, see Section 5.2.50, "MatchURL," on page 112.</p>
MessageBox	<p>Use the <code>MessageBox</code> command to display a dialog box that contains the text specified in the <code><Data></code> variable. The application definition is suspended until the user reacts to this message. The <code>MessageBox</code> can take any number of text arguments, including variables, (for example <code>MessageBox "The user " \$Username " has just been logged onto the system"</code>).</p> <p>For detailed information, see Section 5.2.51, "MessageBox," on page 113.</p>
Multiply	<p>Use to multiply one number by another. You can hard-code the numbers into the application definition, or you can use variables. The results can be output to another variable, or to one of the original numbers.</p> <p>For detailed information, see Section 5.2.52, "Multiply," on page 115.</p>

Command	Description
OnException/ ClearException	<p>Use the <code>OnException</code> command to detect when certain conditions are met. Currently, this is when <i>Cancel</i> is clicked on either of two dialog boxes. When the condition is met, a subroutine is run. Use the <code>ClearException</code> command to reset the exceptions value.</p> <p>For detailed information, see Section 5.2.53, “OnException/ClearException,” on page 116.</p>
Parent/EndParent	<p>Use the <code>EndParent</code> command to terminate a <code>Parent</code> block and set the subject of the application definition back to the original window. You can nest the <code>Parent</code> command, thereby allowing the <code>Parent</code> block to act on the parent of the parent.</p> <p>For detailed information, see Section 5.2.54, “Parent/EndParent,” on page 121.</p>
PickListAdd	<p>Use the <code>PickList</code> command to allow users with multiple accounts for a particular system to choose the account to which they will log in.</p> <p>For detailed information, see Section 5.2.55, “PickListAdd,” on page 123.</p>
PickListDisplay	<p>Use the <code>PickListDisplay</code> command to display the pick list entries built by previous calls to <code>PickListAdd</code>. The <code>PickListDisplay</code> command returns the result in a <code><?Variable></code> sent to the command.</p> <p>For detailed information, see Section 5.2.56, “PickListDisplay,” on page 125.</p>
PositionCharacter	<p>Use the <code>PositionCharacter</code> command in a password policy application definition to enforce that a certain character in the password is a numeral, uppercase, lowercase, or a punctuation character.</p> <p>For detailed information, see Section 5.2.57, “PositionCharacter,” on page 126.</p>
PressInput	<p>Used within a site block to simulate a keyboard enter event.</p> <p>For detailed information, see Section 5.2.58, “PressInput,” on page 127.</p>
ReadText	<p>Use the <code>ReadText</code> command to run in both Windows and Terminal Launcher application definitions. Although the usage and arguments for the use of <code>ReadText</code> with Windows and Terminal Launcher are different, the results of each command are the same.</p> <p>For detailed information, see Section 5.2.59, “ReadText,” on page 128.</p>
RegSplit	<p>Use the <code>RegSplit</code> command to split a string by using a regular expression. <code><Output-String1></code> and <code><Output-String2></code> contain the first and second subexpressions.</p> <p>For detailed information, see Section 5.2.60, “RegSplit,” on page 131.</p>
ReLoadPlat	<p>Use to set the current platform to the last one chosen by the application definition, or if a platform is not chosen, leaves the platform unset.</p> <p>For detailed information, see Section 5.2.61, “ReLoadPlat,” on page 132.</p>
Repeat/EndRepeat	<p>Use the <code>Repeat</code> command to establish an application definition block similar to the <code>If</code> command. The <code>repeat</code> block is terminated by an <code>EndRepeat</code> command. Alternatively, you can use the <code>Break</code> or <code>EndScript</code> commands to break out of the loop.</p> <p>For detailed information, see Section 5.2.62, “Repeat/EndRepeat,” on page 134.</p>

Command	Description
RestrictVariable	<p>Use the <code>RestrictVariable</code> command to monitor a <code><Variable></code> and enforce a specified <code><Password-Policy></code> on the <code><Variable></code>. Any variable specified must match the policy or it is not saved.</p> <p>For detailed information, see Section 5.2.63, "RestrictVariable," on page 135.</p>
Run	<p>Use the <code>Run</code> command to launch the program specified in <code><Command></code> with the specified optional [<code><Arg1></code> [<code><Arg2></code>] ...] arguments.</p> <p>For detailed information, see Section 5.2.64, "Run," on page 138.</p>
Select	<p>Use the <code>Select</code> command to select entries from a combo box or list box control.</p>
SelectListBoxItem	<p>Use the <code>SelectListBoxItem</code> command to select entries from a list box.</p> <p>For detailed information, see Section 5.2.66, "SelectListBoxItem," on page 140.</p>
SelectOption	<p>Use the <code>SelectOption</code> command to select or deselect options within a list box or combo dialog box.</p> <p>For detailed information, see Section 5.2.67, "SelectOption," on page 140</p>
SendKey	<p>Use the <code>SendKey</code> command to work only with Generic and Advanced Generic emulators. You can use the <code>SendKey</code> command in the same manner as the <code>Type</code> command. Generally, the <code>Type</code> command is the preferred command to use. The <code>Type</code> command places the text into the clipboard, and then pastes it into the emulator screen. The <code>SendKey</code> command enters the text directly into the emulator screen.</p> <p>For detailed information, see Section 5.2.68, "SendKey," on page 141.</p>
Set	<p>Use the <code>Set</code> command to copy the value of <code><Data></code> into <code><Variable></code>. The <code><Data></code> can be any text, or another variable, whereas the <code><Variable></code> must be either a <code>?Variable</code> or <code>\$Variable</code>.</p> <p>For detailed information, see Section 5.2.69, "Set," on page 142.</p>
SetCheckBox	<p>Use the <code>SetCheckBox</code> command to select or clear a check box.</p> <p>For detailed information, see Section 5.2.70, "SetCheckBox," on page 143.</p>
SetCursor	<p>Use the <code>SetCursor</code> command to set the cursor to a specified <code><ScreenPosition></code> or <code><X Co-ordinate></code> <code><Y Co-ordinate></code>.</p> <p>For detailed information, see Section 5.2.71, "SetCursor," on page 144.</p>
SetFocus	<p>Use the <code>SetFocus</code> command to set the keyboard focus to a specified <code><#Ctrl-ID></code>.</p> <p>For detailed information, see Section 5.2.72, "SetFocus," on page 145.</p>
SetPlat	<p>By default, variables are stored directly against the platform or application on which you have SecureLogin enabled. For example, if you enable <code>Groupwise.exe</code>, the Groupwise® credentials are stored against the <code>Groupwise.exe</code> platform. <code>SetPlat</code> sets the platform or application from which variables are read and saved.</p> <p>For detailed information, see Section 5.2.73, "SetPlat," on page 146.</p>

Command	Description
SetPrompt	<p>Use the <code>SetPrompt</code> command to customize the text in the Enter SecureLogin Variables dialog boxes. These dialog boxes are used to prompt the user for new variables. You can also use the <code>DisplayVariables</code> command to customize the prompt text in the dialog box (for previously stored variables).</p> <p>For detailed information, see Section 5.2.74, "SetPrompt," on page 149.</p>
Site/Endsite	<p>Begins and ends an application definition, in place of <code>Dialog/EndDialog</code>.</p> <p><code>Site/Endsite</code> are Web commands added to allow for finer control of site matching. More detailed information within a loaded Web site can now be matched upon an used to execute blocks of scripting commands.</p> <p>For detailed information, see Section 5.2.76, "Site/Endsite," on page 151</p>
StrCat	<p>Use the <code>StrCat</code> command to append a second data string to the first data string. For example, <code>StrCat ?Result "SecureRemote " "\$Username"</code>.</p> <p>For detailed information, see Section 5.2.77, "StrCat," on page 153.</p>
StrLength	<p>Use the <code>StrLength</code> command to count the number of characters in a variable and output that value to the destination variable.</p> <p>For detailed information, see Section 5.2.78, "StrLength," on page 154.</p>
StrLower	<p>Use the <code>StrLower</code> command to modify a variable so that all the characters are lowercase.</p> <p>For detailed information, see Section 5.2.79, "StrLower," on page 155.</p>
StrUpper	<p>Use the <code>StrUpper</code> command to modify a variable so that all the characters are uppercase.</p> <p>For detailed information, see Section 5.2.80, "StrUpper," on page 155.</p>
Sub/EndSub	<p>Use the <code>Sub/EndSub</code> commands around a block of lines within an application definition to denote a subroutine.</p> <p>For detailed information, see Section 5.2.81, "Sub/EndSub," on page 156.</p>
Submit	<p>Use the <code>Submit</code> command only in Web application definitions, and only with Internet Explorer, to allow for enhanced control of how and when a form is submitted. The <code>Submit</code> command performs a Submit on the form in which the first password field is found. The <code>Submit</code> command is ignored if used with Netscape.</p> <p>For detailed information, see Section 5.2.82, "Submit," on page 157.</p>
Subtract	<p>Use the <code>Subtract</code> command to subtract one value from another. This is useful if you are implementing periodic password change functionality for an application. You can use the <code>subtract</code> command (in conjunction with the <code>Divide</code> function and the <code>Slna</code> DLL) to determine the number of days that have elapsed since the last password change. Other numeric commands include <code>Add</code>, <code>Divide</code>, and <code>Multiply</code>.</p> <p>For detailed information, see Section 5.2.83, "Subtract," on page 158.</p>
Tag/EndTag	<p>Use the <code>Tag/EndTag</code> commands to find HTML tags.</p> <p>For detailed information, see Section 5.2.84, "Tag/EndTag," on page 160.</p>

Command	Description
TextInput	<p>Use within a site block to input text into a special field.</p> <p>For detailed information, see Section 5.2.85, "TextInput," on page 160</p>
Title	<p>Use the <code>Title</code> command to retrieve the title of a window and compare it against the string specified in the <code><Window-Title></code> argument. For this block of the application definition to run, the retrieved window title and the <code><Window-Title></code> argument must match the text supplied to the <code>Title</code> command in the dialog block.</p> <p>For detailed information, see Section 5.2.86, "Title," on page 161.</p>
Type	<p>Use the <code>Type</code> command to enter data, such as usernames and passwords, into applications. There are reserved character sequences that are used to type special characters, for example TAB and ENTER. If it is not possible to determine Control IDs in a Windows application, and the <code>Type</code> command is not working, use the <code>SendKey</code> command instead.</p> <p>For detailed information, see Section 5.2.87, "Type," on page 162.</p>
WaitForFocus	<p>Use the <code>WaitForFocus</code> command to suspend the running of the application definition until the <code><#Ctrl-ID></code> has received keyboard focus, or the <code><Repeat-Loops></code> expire. The <code><Repeat-Loops></code> is an optional value that defines the number of loop cycles to run. The <code><Repeat-Loops></code> value defaults to 3000 loops if nothing is set. After focus is received, the application definition continues.</p> <p>For detailed information, see Section 5.2.89, "WaitForFocus," on page 167.</p>
WaitForText	<p>Use the <code>WaitForText</code> command so the Terminal Launcher waits for the specified <code><text></code> to display before continuing. For example, the user waits for a username field to display before attempting to type a username.</p> <p>For detailed information, see Section 5.2.90, "WaitForText," on page 169.</p>

Application Definition Language: an Overview

2

The capability of Novell® SecureLogin to create proprietary application definitions is a powerful feature. This application definition command language facilitates single sign-on of all types of applications.

SecureLogin implements application definition commands to provide a flexible single sign-on and monitoring environment. For example, the SecureLogin Windows Agent watches for application login boxes. When a login box is identified, the agent runs an application definition to enter the username, password, and background authentication information.

This section contains the following information:

- ♦ [Section 2.1, “Using Application Definitions,” on page 21](#)
- ♦ [Section 2.2, “Advantages of Using Application Definitions,” on page 21](#)
- ♦ [Section 2.3, “Defining Applications Enabled for Single Sign-On,” on page 22](#)
- ♦ [Section 2.4, “Corporate Definitions,” on page 22](#)
- ♦ [Section 2.5, “What Is an Application Definition?,” on page 22](#)

2.1 Using Application Definitions

You can use application definitions to:

- ♦ Execute the retrieval and entering of correct login details. Application definitions are stored and secured within the directory to ensure maximum security, support for single-point administration, and manageability.
- ♦ Automate many login processes, such as multi-page login and login panels requiring other information that you can store in the directory (such as surname or telephone number).
- ♦ Application definitions can include commands to automate password changes on behalf of users and to request user input when required.
- ♦ Application definitions can accommodate error handling that is generated by the back-end application. For example, handling of invalid logins.

2.2 Advantages of Using Application Definitions

Novell SecureLogin application definitions provide the following advantages:

- ♦ Enables you to define single sign-on methods for almost any Windows, mainframe, Internet, Web, Terminal Server, or UNIX* application.
- ♦ You do not need to install back-end modules on your application servers.
- ♦ Provides the flexibility for you and your application owners to choose what to do once an application generated message is detected, giving you full control over your single sign-on environment.

- ◆ Allows more sophisticated single sign-on to supported applications, including the ability to seamlessly handle several versions of one application. This feature is especially important when you upgrade your applications.
- ◆ Allows you to implement auditing to meet your requirements.
- ◆ SecureLogin data such as user credentials is stored and protected in the directory. The credentials are also stored in the smart card or in the cache; depending on the configuration:
 - ◆ If store credentials on a card are set, then your credentials are on the card and in the directory. The cache file contains only the application definition.
 - ◆ If the cache file is disabled, only the credentials are stored in the directory.
- ◆ On startup, Novell SecureLogin performs the following tasks:
 - ◆ Locates these objects in the directory.
 - ◆ Caches their encrypted contents in memory (and optionally on disk) for later use by the workstation's SecureLogin agent.

2.3 Defining Applications Enabled for Single Sign-On

Novell SecureLogin provides the option to define which applications are enabled for single sign-on. This option gives you:

- ◆ Full control for deciding which applications need to be enabled for single sign-on.
- ◆ The ability to update the entire directory database with a new application login application definition by updating a single object.

2.4 Corporate Definitions

Corporate applications allow scripts to flow down to all users located within a container, allowing central administrators and maintenance of the script.

Corporate application definitions are stored in a Container object rather than on the individual User objects. For users, the result is a less complex system.

For you as the administrator, the improved login mechanisms provide the following:

- ◆ A greater level of accountability with increased productivity and security.
- ◆ A reduced workload at the help desk because of significantly fewer password resets.

2.5 What Is an Application Definition?

An application definition is essentially a list of instructions that SecureLogin follows in order to perform various tasks on various windows. For example, for a Windows application (*.exe), an application definition is written for each executable file that you want SecureLogin to act upon. In that application definition, you are able to assign different instructions to each dialog box or screen that the executable file or application might produce. By doing this, you have the choice of acting upon only the login panel, only selected windows, or every window that is produced by the executable file, such as account locked, invalid username, invalid password, back-end database is down, password expiry, and so on.

SecureLogin follows the application definition from left to right, top to bottom. However, with the use of flow control commands, such as `Call`, it is possible to skip, repeat, or jump to certain parts of the application definition.

2.5.1 Using Dialog Specifier Commands

With the use of `Dialog Specifier` commands, it is possible to assign individual sections of an application definition to the different windows an executable file might produce. This allows the login dialog box, for example, to be treated differently from the Error Message box and so on.

Currently, there are 65 different commands in the Novell SecureLogin application definition language. Many of the SecureLogin commands such as `Repeat` and `Dialog`, have one or two commands that are used to close them.

2.5.2 Reading from and Writing to Variables

Application definition commands have the capability to read from and write to variables. These variables enable SecureLogin to use corporate application definitions, while each individual user's secrets are securely stored in the directory. It is also possible to read attributes, such as the user's full name and phone number, from attributes in the directory.

SecureLogin is not only able to write information to the screen, but is also able to read from it with the use of commands such as `ReadText`. This can be used to extract usernames, domains in use, error messages, and other useful information. `Variable Manipulator` commands can then be used to perform calculations, break apart information, and join it back together again.

All these features come together to form an extremely powerful language that is able to accomplish almost any task that is required.

Using Characters Interpretable by Novell SecureLogin

Using interpretable characters in Novell SecureLogin application definitions has implications for definitions that are created in, or copied from, and pasted from a Microsoft Word.

For example, when you are writing an application definition that requires a “-” (dash) in the command syntax, make sure you use a short “-” or en dash (Unicode glyph U+2013 (Hex) or 8211(Decimal) and cannot be an extended “—” or em dash as generated in Microsoft Word.

In Microsoft Word, when you type a space and one or two hyphens between text, Microsoft Word automatically inserts an ASCII dash or en dash (-). If you type two hyphens and do not include a space before the hyphens, an em dash (—) is created.

Similarly, when you are writing an application definition that requires quotation mark in the command syntax, make sure you use a straight quotation mark (Unicode glyph U+0022 (Hex) or 0034 (Decimal) or the ASCII printable character 34). For quotation mark syntax example, see [Section 4.2.9, “Quotation Marks,” on page 50](#).

In Microsoft Word, when you type a question mark, Word automatically changes straight quotation marks to curly (or smart) quotes, as you type unless the Word *AutoCorrect*, *AutoFormat As You Type* features are disabled.

Managing Application Definitions

3

Application definitions are generally imported, built, or modified in the Management utility of Novell® SecureLogin, tested locally, and then copied to the relevant container, or the organizational unit in multi-user directory environments. Application definitions are imported and exported in the XML file format for ease of distribution and deployment.

This section contains the following information:

- ♦ [Section 3.1, “Application Definition Checklist,” on page 25](#)
- ♦ [Section 3.2, “Exporting and Importing Predefined Applications and Application Definitions,” on page 25](#)
- ♦ [Section 3.3, “Modifying Predefined Applications and Application Definitions,” on page 30](#)
- ♦ [Section 3.4, “Windows Application Definition Tools,” on page 36](#)
- ♦ [Section 3.5, “Application Definition Elements,” on page 40](#)

3.1 Application Definition Checklist

When you have built or modified your application definitions, it is recommended that you test each supported application or the Web page for the following scenarios:

- ♦ Entering a correct username or password.
- ♦ Entering an incorrect username or password.
- ♦ Cancelling a login by the user.
- ♦ Exceeding maximum password retries.
- ♦ A user changing his or her own password.
- ♦ Attempting to change to an illegal password.
This illegal password action is relevant when you define a password policy and you try to define a password that does not match the policy.
- ♦ An administrator cancelling a password change.
- ♦ An administrator changing a user password.
- ♦ Expiry of user password.
- ♦ Locking out the account.
- ♦ Locking out someone from the account.

3.2 Exporting and Importing Predefined Applications and Application Definitions

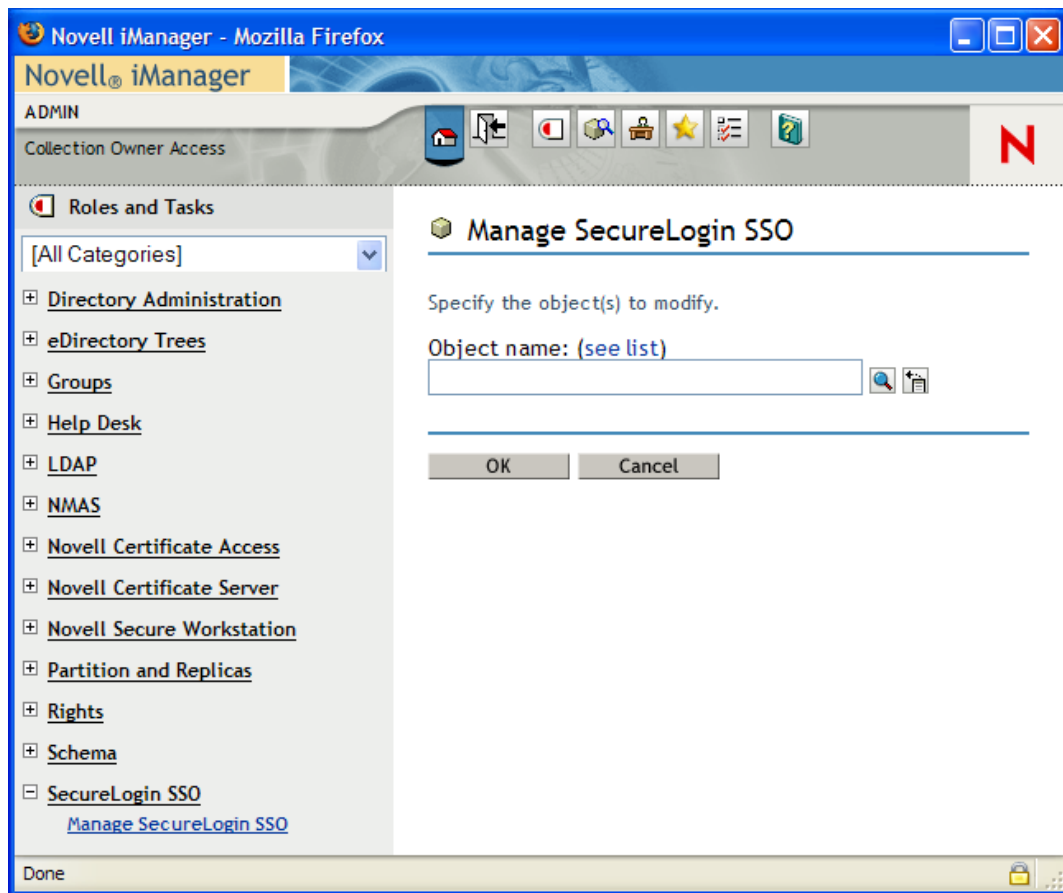
Novell SecureLogin provides export functionality to facilitate distribution of predefined applications and application definitions. Converting predefined applications and application definitions to XML format allows you to distribute and deploy predefined applications and application definitions across directories, software, and hardware platforms.

This section contains the following information:

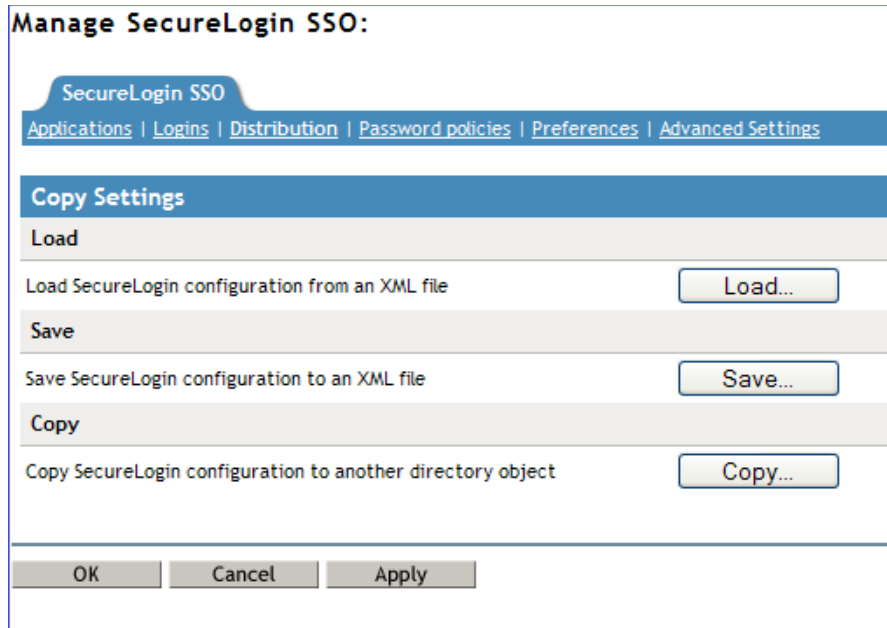
- ♦ “Exporting Individual Applications” on page 26
- ♦ Section 3.2.2, “Importing Individual Applications,” on page 29

3.2.1 Exporting Individual Applications

- 1 Log in to iManager.
- 2 Select *Securelogin SSO* > *Manage Securelogin SSO*. The Manage SecureLogin SSO page is displayed.
- 3 In the object field, specify your object name, then click *OK*.



- 4 Click *Distribution*. The distribution details are displayed.



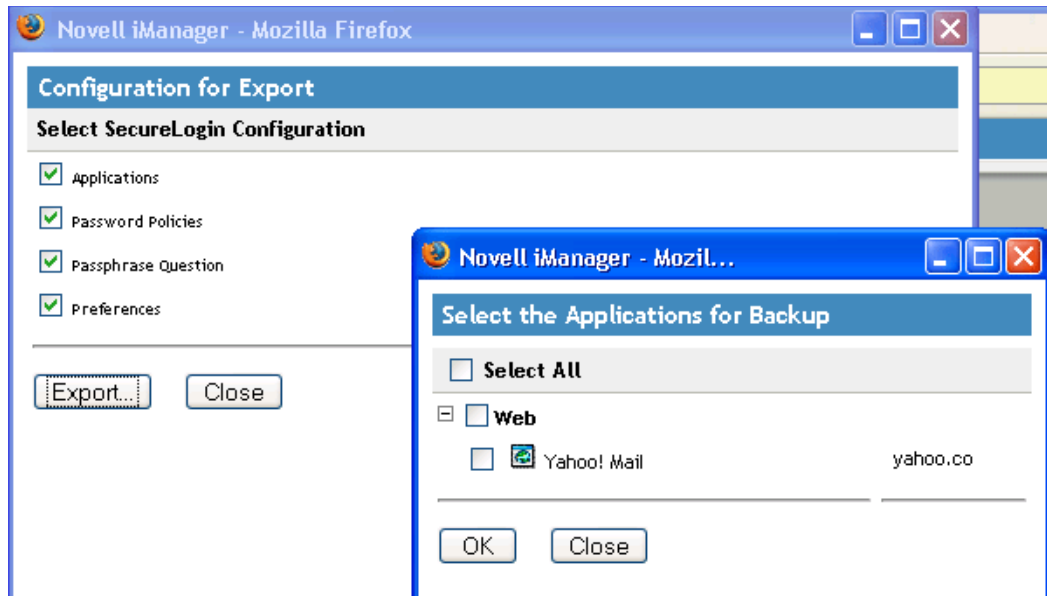
- 5 Click *Save*. The Configuration for Export dialog box is displayed.
- 6 Under *Select SecureLogin Configuration*, select the appropriate text boxes.



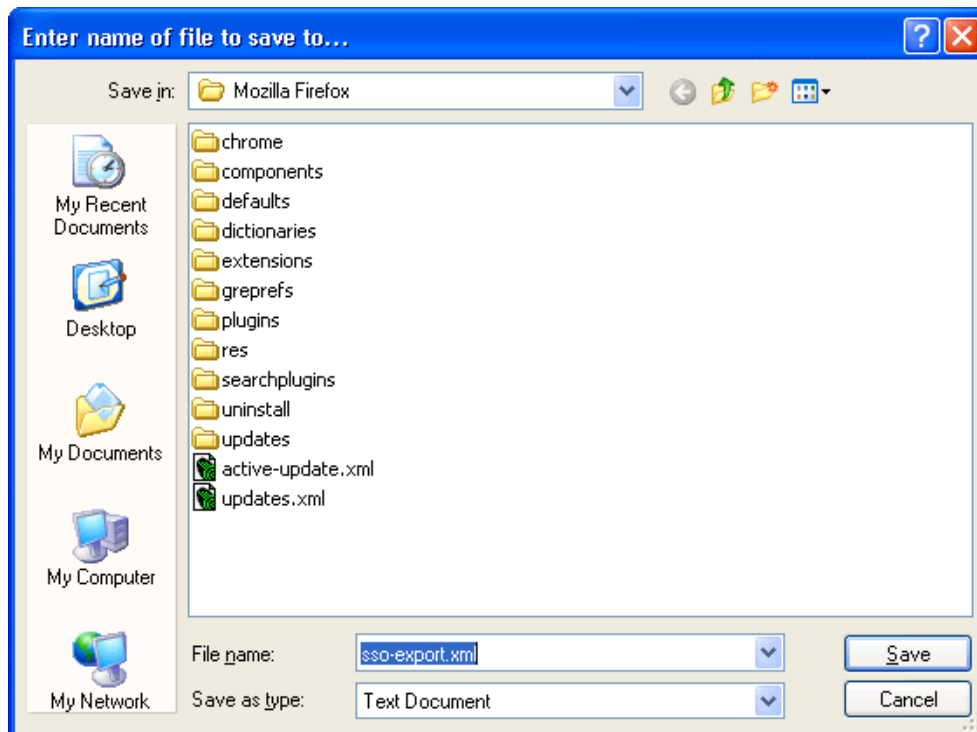
Configuration	Function
Application	Copies, exports, or imports all configured application definitions as displayed in the <i>Application</i> pane.
Credentials	Copies, exports, or imports all credentials as displayed in the <i>Logins</i> pane, excluding passwords for copy settings and unencrypted export or import.
Password Policies	Copies, exports, or imports password policies as displayed in the <i>Password Policies Properties</i> table.
Preferences	Copies, exports, or imports preferences manually set in the <i>Preferences Properties</i> tables.

- 7 Click *Export*. The Select the Applications for Backup page is displayed.

- 8 Select the applications you want to backup.



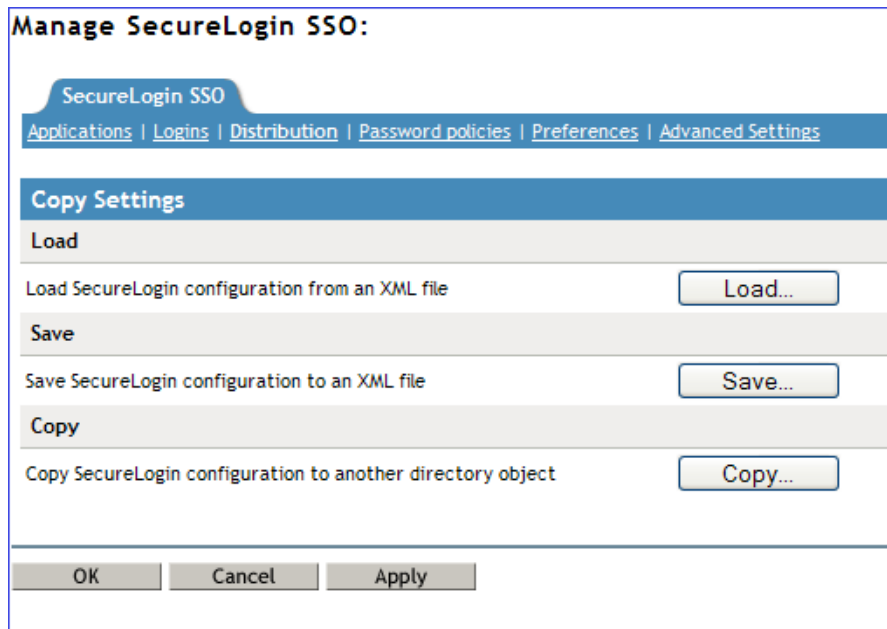
- 9 Click *OK*. The Save File As dialog box is displayed.
10 Provide a name to the file, select the file location, and click *Save*.



NOTE: The file is saved in an XML format.

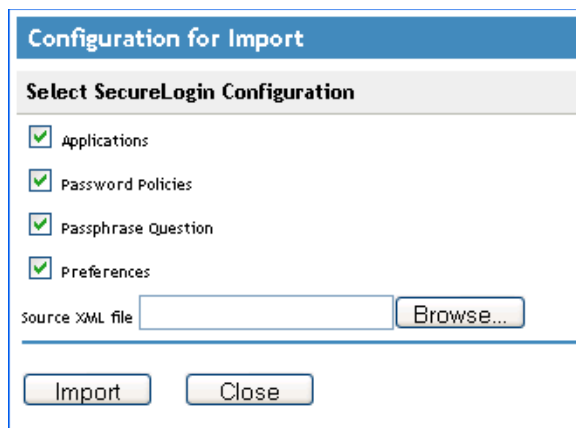
3.2.2 Importing Individual Applications

- 1 Log in to iManager.
- 2 Select *Securelogin SSO > Manage Securelogin SSO*. The Manage SecureLogin SSO page is displayed.
- 3 In the object field, specify your object name, then click *OK*.
- 4 Click *Distribution*. The Distribution details are displayed.



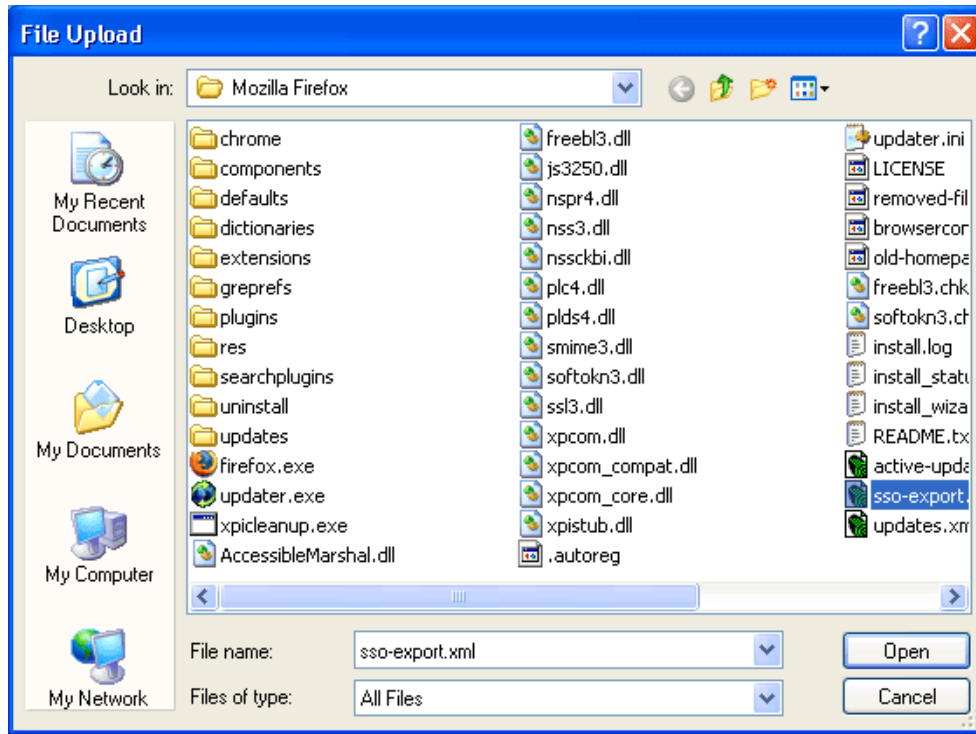
The screenshot shows the "Manage SecureLogin SSO" dialog box. It has a title bar "Manage SecureLogin SSO:" and a tab "SecureLogin SSO". Below the tab is a navigation bar with links: "Applications", "Logins", "Distribution", "Password policies", "Preferences", and "Advanced Settings". The main content area is divided into three sections: "Copy Settings", "Load", and "Save". The "Load" section contains the text "Load SecureLogin configuration from an XML file" and a "Load..." button. The "Save" section contains the text "Save SecureLogin configuration to an XML file" and a "Save..." button. The "Copy" section contains the text "Copy SecureLogin configuration to another directory object" and a "Copy..." button. At the bottom of the dialog are three buttons: "OK", "Cancel", and "Apply".

- 5 Click *Load*. The Select SecureLogin Configuration dialog box is displayed.



The screenshot shows the "Configuration for Import" dialog box. It has a title bar "Configuration for Import" and a section "Select SecureLogin Configuration". Below this section are four checked checkboxes: "Applications", "Password Policies", "Passphrase Question", and "Preferences". Below the checkboxes is a text field labeled "source XML file" and a "Browse..." button. At the bottom of the dialog are two buttons: "Import" and "Close".

- 6 Browse to and select the exported XML file.



- 7 Click *Open* to select the file.

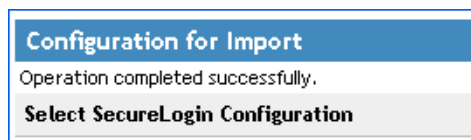
The selected predefined applications and application definitions are copied across to the receiving organizational unit or container.

The selected Securelogin configuration is copied across to the receiving object.

If predefined applications and application definitions currently exist in the receiving object, a confirmation message is displayed to confirm or reject overwrite with the imported data.

- 8 Click *Import* to confirm or click *Cancel* to reject overwriting with the imported data.

A SecureLogin message is displayed to confirm SecureLogin data is loaded.



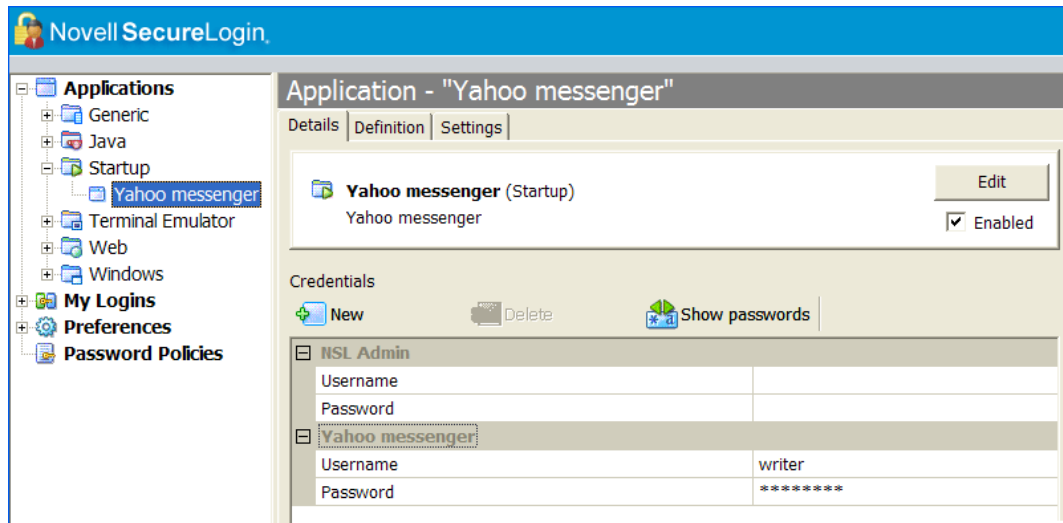
3.3 Modifying Predefined Applications and Application Definitions

Novell SecureLogin predefined applications and application definitions are easily modified to cater to your organization's requirements.

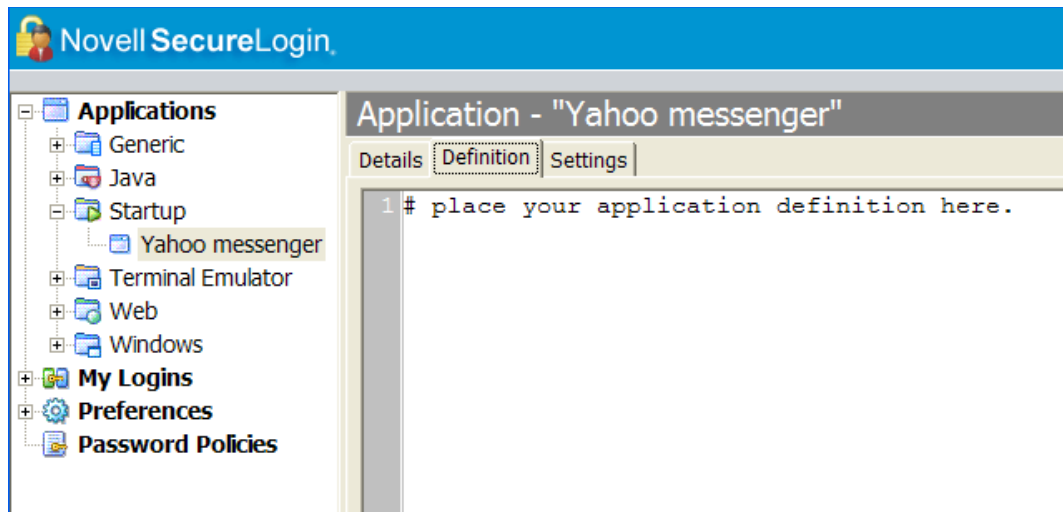
Use the following procedure to modify a Novell SecureLogin predefined application or application definition:

- 1 Double-click the SecureLogin icon in the notification area to display the Personal Management utility.
- 2 Click *Applications*. The Applications pane is displayed.

- 3 Double-click the required application definition. The application details are displayed.



- 4 Select the *Definition* tab. The application definition editor is displayed.



- 5 Modify the application definition or the predefined application, as required.

It is a good practice to include the date and a description of the changes made for future reference.

The predefined Web applications such as eBay or Hotmail under the *Type* drop-down list are titled *Web* and not *Advanced Web*. There is no difference between a *Web* application definition or an *Advanced Web* application definition.

- 6 Click *OK* to save changes and close the Personal Management utility.

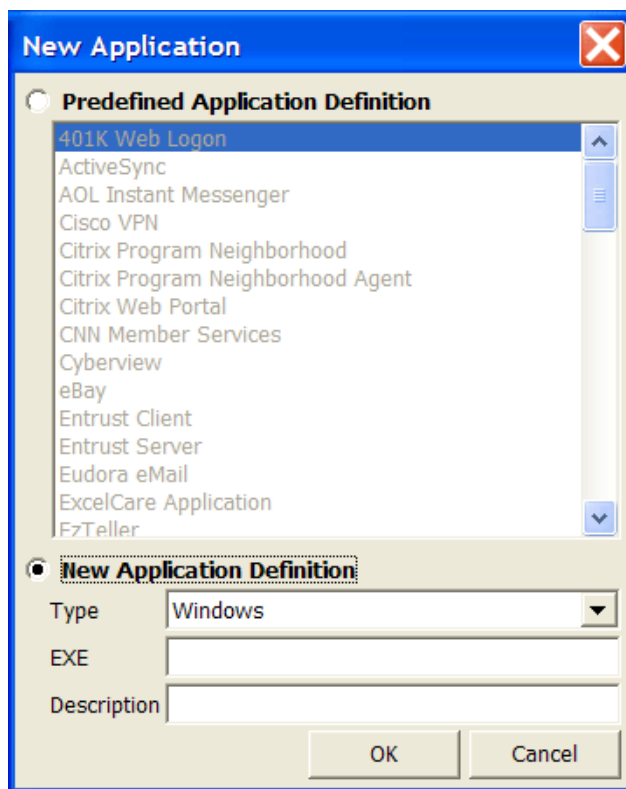
For information on how to modify specific functions see, [Chapter 5, "Command Reference," on page 53](#).

3.3.1 Building an Application Definition in the Personal Management Utility

This section describes how to create and modify SecureLogin application definitions in the Personal Management utility. It is recommended that you test the application definitions locally and then copy them to the relevant container or organizational unit in multi-user directory environments.

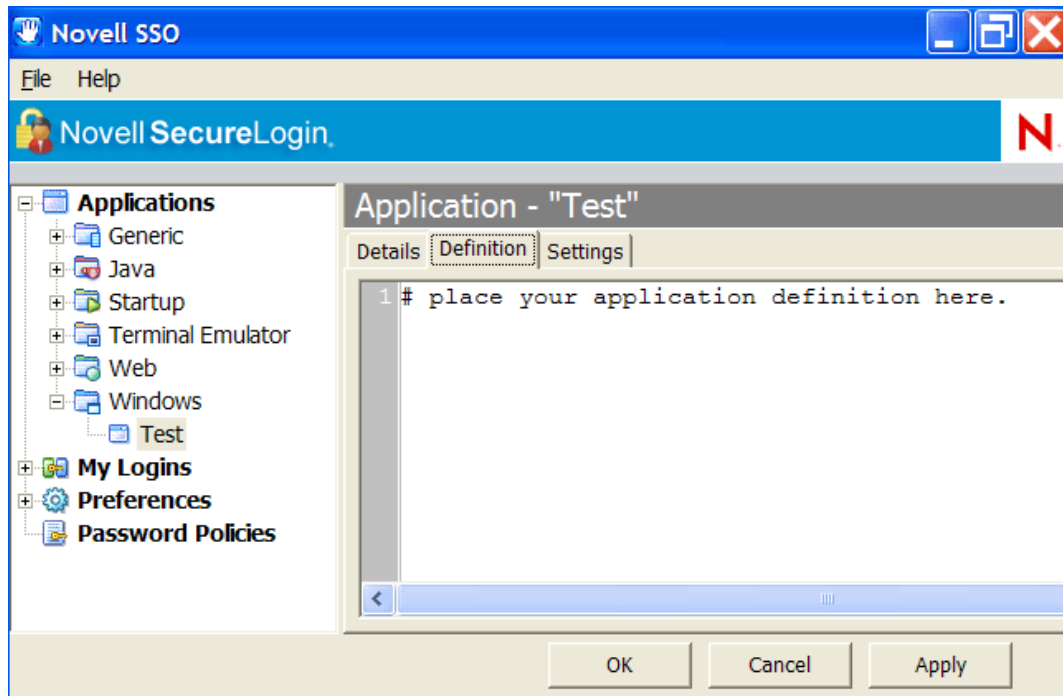
Use the following procedure to create an application definition for a Windows application:

- 1 Double-click the SecureLogin icon in the notification area to display the Personal Management utility.
- 2 Select *File > New > Application*. The New Application dialog box is displayed.
- 3 Click *New Application Definition*, and select the required application type from the *Type* drop-down list.



- 4 Specify other details such as the EXE or the description.
These fields vary based on the application definition type that you have selected. For example, if you select *Windows* as the *Type*, you must fill in the *EXE* and *Description* fields.
- 5 Click *OK*. The application definition is added to the left pane under applications and the details display in the right pane.
- 6 Select *Definition*, and delete the text, # place your application definition here.

Figure 3-1 The Definitions Pane

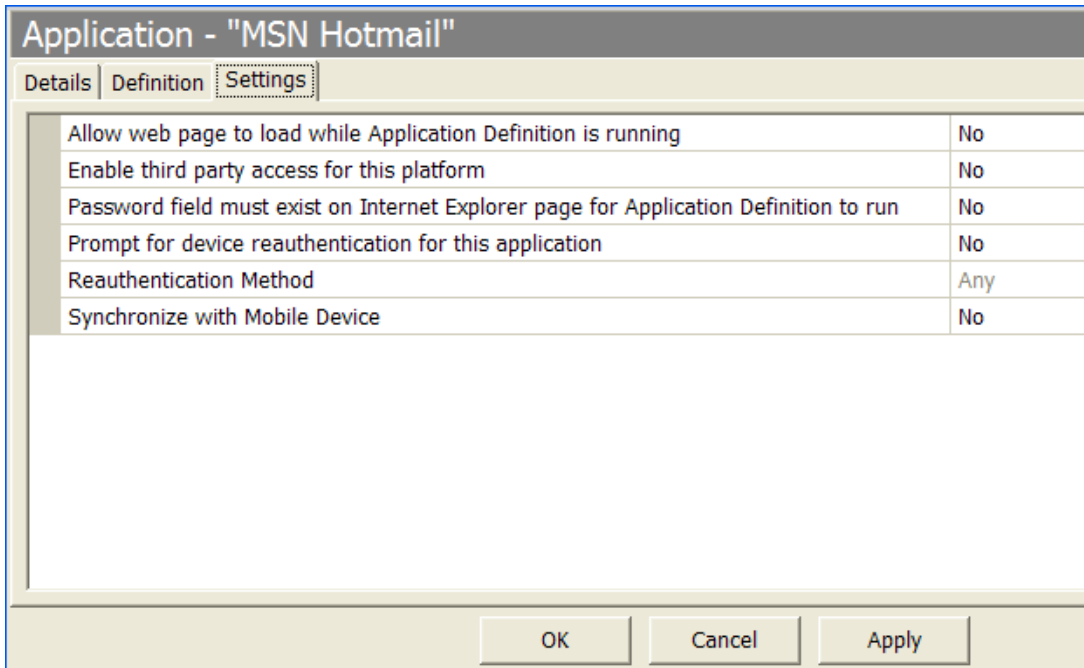


- 7 Specify your application details, then click *OK* to save the changes and close the Personal Management utility.

NOTE: If you are creating multiple application definitions, click *Apply* to save changes without closing the Personal Management utility.

Settings Tab

Figure 3-2 *The Settings Options*



The *Settings* tab includes the following options for application definitions and predefined applications:

Table 3-1 *Settings Options*

Option	Description
<i>Allow web page to load while application definition is running</i>	<p>Applies to Microsoft Internet Explorer and application definitions created for Web pages and JavaScript* login that execute in a Web page.</p> <p>By default, this option is set to <i>No</i>. This suspends completion of any other Internet Explorer tasks until the log in is completed.</p> <p>If this option is set to <i>Yes</i>, Novell SecureLogin allows Internet Explorer to continue functioning while Novell SecureLogin is executing the login.</p>
<i>Enable third party access for this platform</i>	<p>By default, this option is set to <i>No</i>. This disables the API access for this predefined application or the application definition.</p> <p>If this option is set to <i>Yes</i>, it disables the API access for this predefined application or application definition.</p>

Option	Description
<i>Password field must exist on Internet Explorer page for application definition to run</i>	<p data-bbox="812 260 1336 344">Applies to Microsoft Internet Explorer and application definitions created for Web pages and JavaScripts within Web pages.</p> <p data-bbox="812 369 1336 453">If this option is set to <i>Yes</i>, Novell SecureLogin does not execute automated login for pages without a password field.</p> <p data-bbox="812 478 1336 590">If this option is set to <i>No</i>, your Web application returns errors on pages without password fields that you need to handle with Novell SecureLogin. For example, password change successful.</p>
<i>Prompt for device reauthentication for this application</i>	<p data-bbox="812 615 1336 672">Allows you to reauthenticate an application against an Advanced Authentication (AA) device.</p> <p data-bbox="812 697 1336 781">By default, this option is set to <i>No</i>, which means that users are not prompted for device reauthentication for the application.</p> <p data-bbox="812 806 1336 856">If this option is set to <i>Yes</i>, user are prompted for device reauthentication for the application.</p>
<i>Reauthentication Method</i>	<p data-bbox="812 882 1336 993">This option allows you to reauthenticate an application against an SLAA device where Novell SecureLogin is used in along with SLAA or the NMAS™ infrastructure.</p> <p data-bbox="812 1018 1336 1102">This option is available only when <i>Prompt for device reauthentication for this application</i> is set to <i>No</i>.</p> <p data-bbox="812 1127 1276 1157">The reauthentication methods available are:</p> <ul data-bbox="836 1182 1068 1451" style="list-style-type: none"> ◆ Any ◆ Biometric ◆ Smart card ◆ Token ◆ Password ◆ Passphrase ◆ Directory password
<i>Synchronize with Mobile Device</i>	<p data-bbox="812 1476 1336 1587">This option is set to <i>No</i> by default, enabling synchronization to an API-enabled hand-held device, for this predefined application or application definition.</p> <p data-bbox="812 1612 1336 1728">If this option is set to <i>Yes</i>, it disables synchronization to an API-enabled handheld device for this predefined application or application definition.</p>

3.4 Windows Application Definition Tools

Novell SecureLogin provides wizards to assist with the creation of basic application definitions. For more complex applications and requirements, Novell SecureLogin provides the following tools to assist with finding the application information required to build an application definition:

- ♦ [Section 3.4.1, “Finding Application Details with Window Finder,” on page 36](#)
- ♦ [Section 3.4.2, “Finding Application Details with the Login Watcher,” on page 38](#)

3.4.1 Finding Application Details with Window Finder

The Novell SecureLogin Window Finder finds windows applications details, including control and dialog box IDs. Novell SecureLogin might require this information to identify specific objects in order to uniquely identify the application.

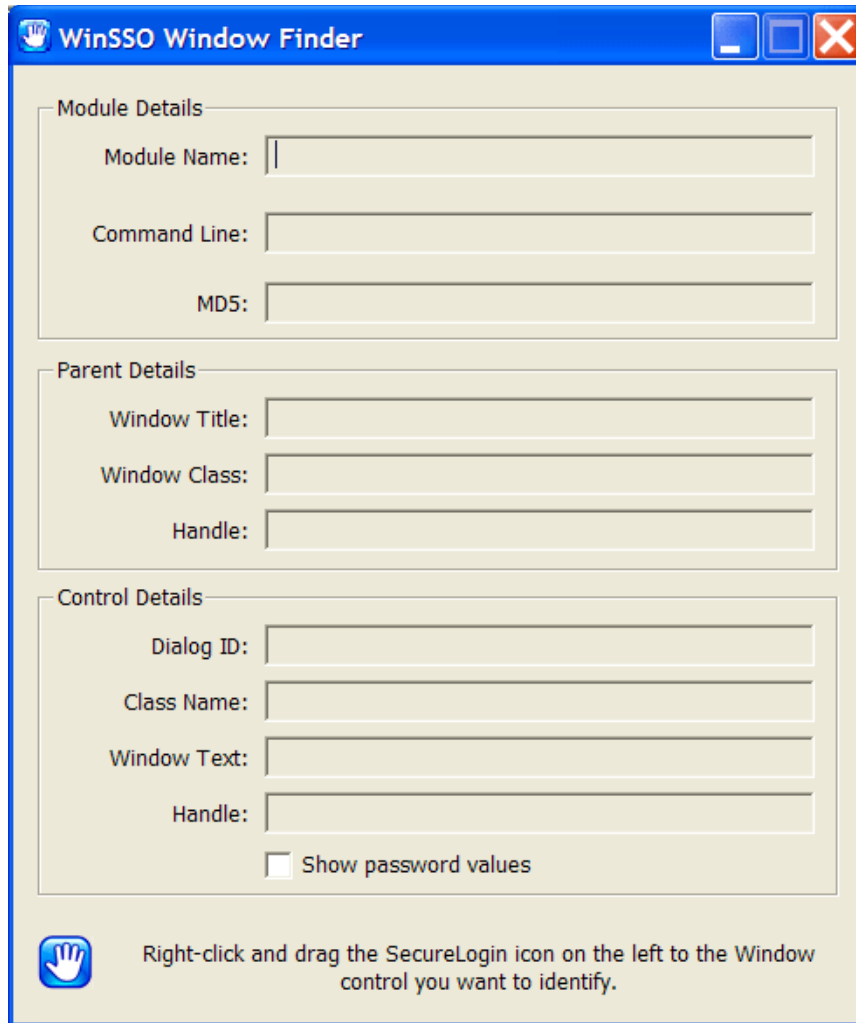
Control IDs are used to uniquely identify objects within a window. Window Finder extracts this information from the application for use in the application definition.


- ♦ [“Starting the Windows Finder” on page 36](#)
- ♦ [“WINSSO Window Finder Details” on page 37](#)

Starting the Windows Finder

The following procedure uses the Novell SecureLogin test application provided on the Novell SecureLogin product installer package or your other distribution source.

- 1 On the Windows *Start* menu, select *All Programs > Novell Securelogin > Window Finder*. The Window Finder is displayed.



- 2 Right-click the Novell SecureLogin  icon in the dialog box, drag it to the required window, field or control, and release the mouse button.

WINSSO Window Finder Details

The following table lists the fields in the WinSSO Window Finder:

Table 3-2 *Window Finder Details*

Field	Description
Module Details Section	
Module Name	This is the Windows executable name for the selected application. This is the application name for a Windows application definition or the predefined application.

Field	Description
Command Line	This is the full command line used to start the application. You can use this information in along with the <code>GetCommandLine</code> command.
Parent Details Section	
Window Title	This is the title of the window of the selected control. Use with the <code>Title</code> command in the <code>Dialog/EndDialog</code> section of the application definition.
Window Class	This is the Windows class name for this dialog or window. Use with the <code>Class</code> command in a <code>Dialog</code> or <code>EndDialog</code> section.
Handle	This is the internal Windows handle for this window. This is generally not used in application definitions.
Control Details Section	
Dialog ID	This is the unique number identifying the control. Use it with various commands, including <code>Type</code> , <code>SetPlat</code> , and <code>Click</code> .
Class Name	This is the Windows class name for the control. Novell SecureLogin supported classes, which include <code>Edit</code> , <code>Combo box</code> , and <code>Static</code> .
Window Text	This is the test that exists on the control. Useful to copy and paste into the application definition editor. <ol style="list-style-type: none"> 1. Note or copy the required details from the WinSSO Window Finder window from the relevant fields. 2. Click <i>Close</i> to quit and close the WinSSO Window Finder window.

3.4.2 Finding Application Details with the Login Watcher

The Login Watcher records login and Windows application data to provide information that you might need for creating an application definition.

- ◆ [“Order Information Is Recorded and Stored” on page 38](#)
- ◆ [“Information Details” on page 39](#)
- ◆ [“SecureLogin Test Application Example” on page 39](#)

Order Information Is Recorded and Stored

Information is recorded and stored in a text file in the following order:

```
Time||Module Name||Window Handle||Window Text||Class Name||Parent||Visible Flag||Title
Flag||Control ID
```

NOTE: The Login Watcher records all log in information, including usernames and passwords, in a text file. This text file might be a security issue.

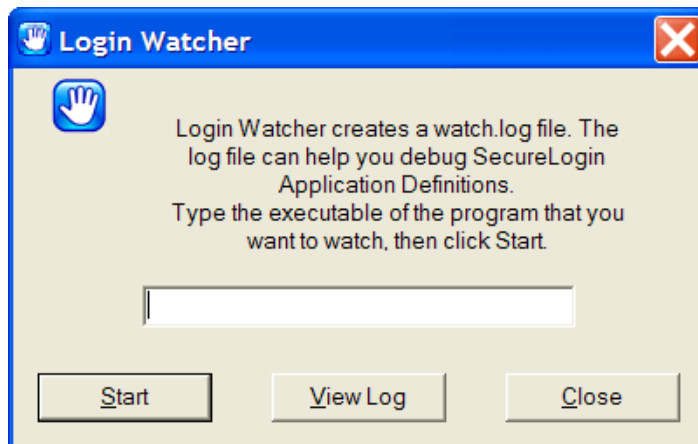
Information Details

Information Item	Description
Time	Milliseconds elapsed since the Login Watcher started.
Module name	Name of the executable being recorded.
Window handle	Unique identifier for the window.
Window text	All text displayed in the window, which includes text entered during login and text displayed as labels for fields and buttons.
Class name	Name of the window class.
Parent	Window handle of the parent window.
Visible flag	Refers to top-level windows that have the style set to <i>Visible</i> . If set to <i>Visible</i> , the word Visible displays; otherwise the field is empty.
Title flag	Refers to top-level windows that have the style set to display the Window Title. If the title is not displayed, then the field is empty.
Control ID	The unique numerical identifier for the windows object.

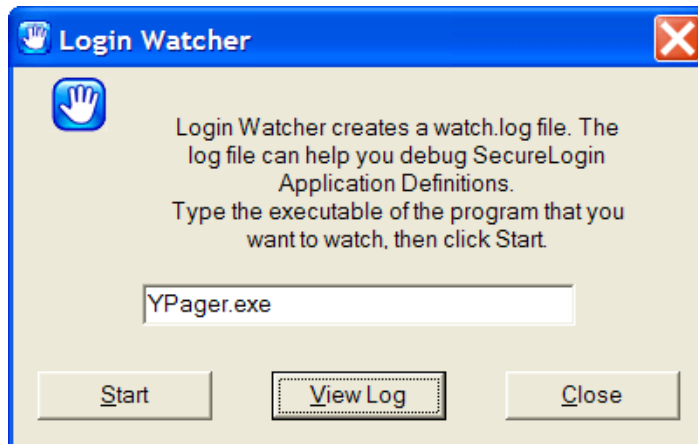
SecureLogin Test Application Example

The following procedure uses the SecureLogin test application:

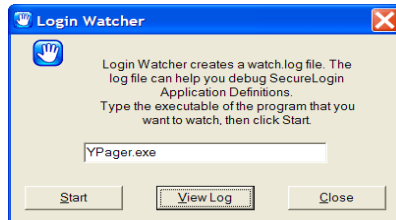
- 1 Right-click the Novell SecureLogin icon on the notification area.
- 2 Select *close* from the menu.
- 3 Right-click the Windows *Start* menu > *Explore*.
- 4 Double-click `loginwatch.exe`, by default located at `<...>\program files\novell\securelogin\tools`. The Login Watcher dialog box is displayed.



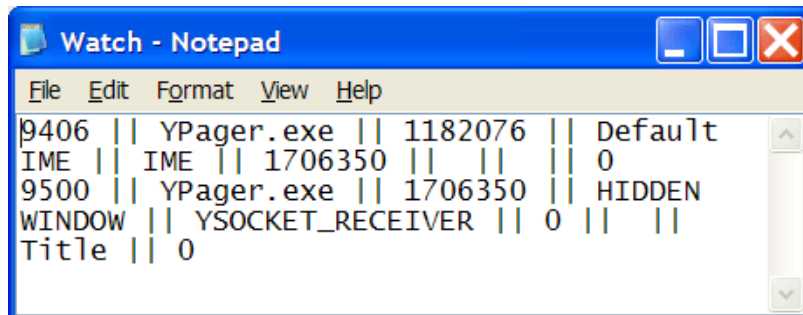
- 5 Specify the executable filename in the Login Watcher field. For example, YPager.exe.



- 6 Click *Start*. The Now Recording Log confirmation dialog box appears.



- 7 Log in to the relevant application.
- 8 Click *Stop* when logged on successfully to return to the Login Watcher dialog box.
- 9 Click *View Log*. Novell SecureLogin starts the Notepad application and displays the watch.txt file with login details recorded.



- 10 Note the required information or save the text file with a different name.
- 11 Click the Login Watcher dialog box. Click *Close*.

3.5 Application Definition Elements

Application definitions use various symbols to define the function of each line. The following table lists the definitions for these symbols.

Table 3-3 *Symbol Definitions*

Symbol	Description
#	<p>Use the number of this symbol to define a line of text as a comment field. Comment fields are used to leave notes.</p> <p>Any line that starts with a # is ignored.</p> <p>Use comment lines for the following:</p> <ul style="list-style-type: none">◆ Defining sections of an application definition, for example the login window and Change Password window.◆ Explaining complex sections.◆ Removing command lines during creation and editing of the application definition. This saves continuously deleting and rewriting lines while testing.◆ Making notes such as when the application definition was written, what version of the software it was written for, and so on. <p>When used as part of a command, such as <code>Class</code> or <code>Type</code>, the symbol specifies a numerical value.</p> <p>You can use these numerical values to specify a target for the command.</p>
" "	<p>Use quotation marks to group together text or variables that contain spaces. Quotation marks are used with commands such as <code>Type</code>, <code>MessageBox</code>, and <code>If -Text</code>.</p> <p>For these command lines to work, you must use quotation marks in the following method to group the text together:</p> <ul style="list-style-type: none">◆ <code>Type "Database 2"</code>◆ <code>MessageBox "Please confirm your log in details."</code>◆ <code>If -Text "Login failure"</code>
\$	<p>Use the dollar sign to define the use of a Novell SecureLogin variable stored in the directory for later use by that user.</p> <p>These variables are used to store information such as usernames and passwords.</p>
?	<p>Use the question mark to define the use of a runtime variable. The values of these variables are not stored in the directory; they are reset each time Novell SecureLogin is started.</p> <p>Alternatively, with the use of the <code>Local</code> command, these variables are reset each time the application definition is started.</p> <p>These variables are used to store temporary information, such as counting, data processing, and date information. The question mark is also used with several internal system generated variables.</p>

Symbol	Description
%	<p>Use the percentage sign to define the use of a directory attribute. The attributes available vary depending on the directory in use, and the setup of the directory.</p> <p>Examples of the attributes you can use are FCN and %Surname.</p> <hr/> <p>NOTE: Quotes are required around the variable if the attribute name contains a space. For example,</p> <pre>Set ?text "%Login Time"</pre> <p>or</p> <pre>Messagebox "%Given Name"</pre> <p>For more information, see Section 4.1.2, "Directory Attribute Variables," on page 43.</p>
!	<p>Use the exclamation mark to define the use of a passticket. A passticket is a one-time password (OTP) that is generated by using a combination of an encryption key, encryption offset, and the current time.</p> <p>Such passwords are only valid for a short period of time (generally between 30 seconds and 2 minutes). You can manually define the encryption key and offset, or the Novell SecureLogin can generate it automatically.</p> <p>If the exclamation mark is included as the first character in a text string, then precede it with a backslash, or Novell SecureLogin attempts to define a passticket. For example, <code>Type \!xyz</code> types "!xyz" to the application.</p>
\	<p>Use the backslash with the <code>Type</code> and <code>SendKey</code> commands to specify the use of a special function.</p> <p>The backslash is used along with values to perform the simulation of pressing keys. Examples of frequently used functions are provided in the following list:</p> <ul style="list-style-type: none"> ◆ Alt-F: Alt+F on the keyboard in Windows and Web applications. ◆ ID: Delete key in a Windows and Web applications. Not applicable to terminal emulators. ◆ IN: Enter key in a Windows and Web applications. Not applicable to terminal emulators. ◆ IT: Tab in Windows and Web applications. ◆ \-T: Shift+Tab in Windows and Web applications.
@	<p>Use the same way as the backslash symbol, except its use is limited to HLLAPI-enabled emulators.</p> <p>This symbol is used along with values to perform the simulation of key presses. For example, use <code>@E</code> to simulate pressing Enter in a terminal emulator application.</p>
-	<p>Use the hyphen as a switch within several commands, such as <code>If</code> and <code>Type</code>.</p> <p>The hyphen is used along with values to modify the behavior of commands (such as <code>-Raw</code>), or switch on or off certain functions (such as <code>-YesNo</code>).</p>

Application Definition Variables

4

This section contains the following information:

- ♦ [Section 4.1, “Types of Variables,” on page 43](#)
- ♦ [Section 4.2, “Application Definition Variables,” on page 47](#)

4.1 Types of Variables

Novell SecureLogin supports the use of four different types of variables:

- ♦ Stored
- ♦ Runtime
- ♦ Directory attribute
- ♦ Passticket

NOTE: Specify variables without spaces, for example \$Username_Alias. If you use spaces you must enclose the entire variable in quotation marks, for example, "\$Username Alias".

4.1.1 Using a Variable to Change the Default Platform

Each variable defaults to the platform specified in the application definition or the predefined application name. You can use a variable to change the platform, for example you can have an application definition named www.website1.com, for example:

```
type $username
type $password password
```

An application definition named www.website2.com might use the variables from www.website1.com, for example:

```
type $username(www.website1.com)
type $password(www.website1.com) password
Directory attribute variables
```

4.1.2 Directory Attribute Variables

Novell SecureLogin reads directory attributes from the currently logged in user's object.

For example:

```
type%cn
```

reads the CN attribute from the currently logged in user's object and specifies it.

You can only use the percentage symbol (%) variables when Novell SecureLogin is configured to use a directory, and only on single-valued text attributes.

Quotes are required around the variable if the attribute name contains a space.

For example:

```
Set ?text "%Login Time"  
MessageBox "%Given Name"
```

4.1.3 Stored Variables

Stored variables are the most common style of variable used in application definitions and Predefined Applications. They are preceded with a dollar symbol (\$). Use these variables to store the values used during the login process, such as usernames, passwords and any other details that are required.

This section contains the following information:

- ♦ [“Storing the Variables” on page 44](#)
- ♦ [“Using Stored Variables” on page 44](#)

Storing the Variables

The values of these variables are stored in the directory under the user object. They are encrypted so that only the user can access them. You can store variables separately for each application definition and predefined application, so the username variable for one application can be different from the username variable for another application. It is, however, possible to set an application to read variables from another application's application definition and predefined application. This is useful for applications that share user accounts or passwords.

For details, see [Section 5.2.73, “SetPlat,” on page 146](#)

Using Stored Variables

If a stored variable is referenced in an application definition and predefined application, and there is no value stored for that variable (for example, the first time the program is run), SecureLogin prompts the user to enter a value for the variable. This is an automatic process. It is also possible to manually trigger this process to prompt a user to enter new values for particular variables.

```
Dialog  
Class #32770  
Title "Log on"  
EndDialog  
Type $Username #1001  
Type $Password #1002  
Click #1
```

NOTE: If you want to hide a variable from an administrator by displaying it as **** instead of clear text, begin the variable name with \$Password.

For example, the \$PasswordPIN variable is protected as described, however \$PIN is not.

For more details, see [Section 5.2.20, “DisplayVariables,” on page 82](#) and [Section 5.2.9, “ChangePassword,” on page 69](#).

4.1.4 Runtime Variables

Runtime variables are generally used for storage of calculations, processing data, and date information. You can also use them for temporary passwords and usernames.

Runtime variables are preceded by the question mark symbol (?). They have two modes:

- ◆ Normal runtime variables are reset each time SecureLogin is started.
- ◆ Local runtime variables are reset each time the application definition and predefined application is started.

Runtime variables are Normal by default. For details on how to switch a runtime variable to Local mode, see [Section 5.2.43, "Local," on page 103](#).

Using Runtime Variables

Runtime variables are not stored in the directory or the Novell SecureLogin cache; they are used straight from the computer's memory. For this reason, it is important not to use runtime variables for the storage of usernames, passwords, or other details Novell SecureLogin will need to access in the future.

If runtime variables are used for such details, the user is prompted to enter them each time the application definition or predefined application is run, or each time Novell SecureLogin is restarted. Users are not prompted for `?variables` that have no value. These variables are given the value `<NOTSET>`.

Example of a Runtime Variable

```
Dialog
Class #32770
Title "ERROR"
EndDialog

Local ?ErrorCount
Increment ?ErrorCount

If ?ErrorCount Eq "2"
MessageBox "This is the second time you have received this
error. Would you like to reset the application?" -YesNo ?Result
If ?Result Eq "Yes"
KillApp "App.exe"
Run "C:\App\App.exe"
Else
Set ?ErrorCount "0"
EndIf
EndIf
```

4.1.5 Passticket Variables

Passticket variables are preceded with the exclamation mark symbol (!). To use a passticket variable, you must create and define numerical values for stored variables with the names `$DESKEY` and `$DESOFFSET`. These numbers are then used by the Novell SecureLogin application definition or the predefined application parser to generate the one-time password.

Using a Passticket Variable to Generate a Password

Once you have defined the stored variables, use the following passticket variable to generate a password.

```
!<Name of application definition>
```

or

```
!default
```

For example, if you want to use a passticket variable for the Microsoft Outlook application, create two stored variables called `$DESKEY` and `$DESOFFSET` under the Outlook application definition. Then, set values for the two stored variables, which allows you to use the variable `"!Outlook"` whenever you need to generate a one time password.

You can also use `"!Default"`, which automatically reads the values from the current application definition.

If the credentials used to generate one time passwords do not already exist in a secured area of the SecureLogin cache (that is, the `$DESKEY` and `$DESOFFSET` variables are not defined), then they are retrieved from the closest SecureLogin Advanced Authentication server. For more information on this, contact Novell Technical Support.

4.1.6 SecureLogin Supported Variables

SecureLogin is able to read details from the system and use them to create variables that you can incorporate into the application definition. These variables are automatically generated as Runtime Variables and used in the same manner within any application definition.

Variable	Description
<code>?SysVersion(system)</code>	The local SecureLogin windows agent version. You can use this variable to determine if specific support is built into the product running on the user's workstation. The version convention is to use two digits for each section read from right to left, and leading zeros are removed. For example, version 3.0.4.0 would be returned as 03000400.
<code>?BrowserType(system)</code>	Contains Internet Explorer or Netscape and indicates in which browser the application definition or predefined application is running. This variable is only set in a Web application definition or predefined application.
<code>?SysUser(system)</code>	The name of the user currently using SecureLogin.
<code>?SysPassword(system)</code>	The directory password of the user currently using SecureLogin. This variable is only available if the appropriate options are chosen when installing SecureLogin.
<code>?SysContext(system)</code>	The context within which the current SecureLogin user's directory object exists.

Variable	Description
?SysTree (system)	<p>The name of the directory tree that SecureLogin is currently using.</p> <hr/> <p>NOTE: The variable ?SysTree returns the Domain name when using the Microsoft GINA (MS-AD or ADAM) and the tree name or port number when using the Novell GINA or LDAP installation.</p>
?SysServer (system)	<p>The name of the server or the IP address of the server that was entered in the Novell Client™ login panel.</p> <hr/> <p>NOTE: This variable is only available if the Novell client login extension is installed (NDS) and is not available if the MS Active Directory (MS-AD) or ADAM option has been installed.</p>
?SysTSLaunched (system)	<p>Contains the condition state value when SLLauncher is run.</p> <p>This variable is set to True when a script is being executed by SLLauncher. Otherwise it will be <NOTSET>.</p>
?CurrTime (system)	<p>The running time in seconds from January 1970 to the present. You can use this variable to force password changes every X days, and so on.</p> <p>Do not use the application definition to force a password change if you want to continue having the application generate the change password event (recommended). Use this variable on applications where you cannot set a password expiry at the application back end.</p>

4.2 Application Definition Variables

The following are some of the best practice rules to follow when creating an application definition. These rules make reading the application definition easier and also help if you need to make modifications in the future.

This section contains the following information:

- ◆ [Section 4.2.1, “Symbols Used,” on page 48](#)
- ◆ [Section 4.2.2, “Capitalization,” on page 48](#)
- ◆ [Section 4.2.3, “Comments,” on page 48](#)
- ◆ [Section 4.2.4, “Switches,” on page 48](#)
- ◆ [Section 4.2.5, “Variables,” on page 49](#)
- ◆ [Section 4.2.6, “Indent Sections,” on page 49](#)
- ◆ [Section 4.2.7, “Blank Line Between Sections,” on page 49](#)
- ◆ [Section 4.2.8, “Writing Subroutine Sections,” on page 50](#)
- ◆ [Section 4.2.9, “Quotation Marks,” on page 50](#)
- ◆ [Section 4.2.10, “Password Policy Names,” on page 50](#)
- ◆ [Section 4.2.11, “Regular Expressions,” on page 51](#)

4.2.1 Symbols Used

Table 4-1 *Description of Symbols*

Symbol	Description
< >	Angle brackets represent an item. For example, text, variable, or value.
[]	Square brackets represent an optional item. If an item is not marked with square brackets, it is a compulsory item.
↵	Indicates a line break

4.2.2 Capitalization

Use capitalization where applicable.

Table 4-2 *Capitalization*

Instead of...	Use...
<code>messagebox "some text" -yesno ?result</code>	<code>MessageBox "Some text" -YesNo ?Result.</code>

4.2.3 Comments

Use comments throughout to explain what each section does and how it does it.

Table 4-3 *Comments*

Instead of...	Use...
<code>Dialog</code>	<code># Written By: B. Smith 07/Jun/2002</code>
<code>Class #32770</code>	<code># Last Modified By: C. Silvagni 13/Mar/2003</code>
<code>Title</code>	<code># Logon Dialog Box</code>
<code>"Log on"EndDialog</code>	<code>Dialog</code> <code>Class #32770</code> <code>Title "Log on"</code> <code>EndDialog</code>

4.2.4 Switches

Switches are placed directly after the command, for example, `Type -Raw, If -Text.`

Table 4-4 *Switches*

Instead of...	Use...
Type \$Username -Raw	Type -Raw \$Username

4.2.5 Variables

All variable names start with a capital letter.

Table 4-5 *Variables*

Instead of...	Use...
Type \$username	Type \$Username

4.2.6 Indent Sections

Indent sections between pairs of commands, for example `Dialog`, `Repeat`, `If`.

An indent of three spaces is recommended.

Table 4-6 *Indent Sections*

Instead of...	Use...
If -Text "Some text"	If -Text "Some text"
#Do thisElse	#Do thisElse
#Do This	#Do this
EndIf	EndIf

4.2.7 Blank Line Between Sections

Leave a blank line between sections, for example, between the `Dialog` Block and the rest of the application definition.

Table 4-7 *Blank Line between Sections*

Instead of...	Use...
# Log on Dialog Box	# Logon Dialog Box
Dialog	Dialog
Class #32770	Class #32770
Title "Log on"	Title "Log on"
EndDialog	EndDialog
Type \$Username #1001	Type \$Username #1001
Type \$Password #1002	Type \$Password #1002
Click #1	Click #1

4.2.8 Writing Subroutine Sections

Write subroutine sections at the bottom of the application definition and not partway through.

The name of the subroutine should describe its function. Do not use a numeric name. The name should follow the capitalization rule.

Wherever possible, use the `Include` command to create generic application definitions for frequently used elements, for example password change procedures. For common processes within the same application definition, use subroutines.

4.2.9 Quotation Marks

Always use quotation marks around segments of text in commands.

Table 4-8 *Using Quotation Marks*

Instead of	Use
Type TextOrIf -Text Login	Type "Text"OrIf -Text "Log on"

4.2.10 Password Policy Names

Password policy names should represent the program they are used for. Do not use numerical names.

Table 4-9 *Password Policy Names*

Instead of...	Use...
PasswordPolicy3	GroupwisePasswordPolicy

At the top of the application definition, enter and comment out information, for example, the author and the date of the last modification.

Table 4-10 *Example*

Instead of...	Use...
Dialog	# Written By: B. Smith 07/Jun/2002
Class #32770 Title "Log on"	# Last Modified By: C. Silvagni 13/Mar/2003
EndDialog	# Logon Dialog Box
	Dialog
	Class #32770
	Title "Log on"
	EndDialog

NOTE: Always place the `Title` command after all other commands in the `Dialog` block.

4.2.11 Regular Expressions

Regular expressions are text patterns normally used for string matching. Regular expressions might contain a mix of plain text and special characters to indicate the kind of matching to be done.

For example, if you are searching for any numeric character, then the regular expression that you use for the search is, "[0-9]".

The square [] brackets indicate that the character that is compared must match any one of the characters enclosed with in the brackets. The dash (-) between the zero (0) and nine (9) indicates that the range is between the number zero and nine.

If you need search for a special character, then you must use the backslash (\) before the special character.

The following table briefly describes the Novell supported special characters that can be used in regular expressions within the Novell SecureLogin application definition, in particular the `RegSplit` command detailed in the [Section 5.2.60, "RegSplit," on page 131](#).

Table 4-11 *Special Characters*

Character	Description
\ (Backslash)	The \ is an escape character indicating that the next character must be used as a regular search character and not as a special character. For example, the regular expression "*" matches a single asterisk and the expression "\\\" matches a single backslash.

Character	Description
^ (Caret)	<p>The ^ is an anchor. If you use the ^ preceding any character, it searches the beginning character of any string.</p> <p>For example, the expression "A^" matches an "A" only at the beginning of the string.</p>
[^ (Square bracket and Caret)	<p>The ^ immediately following [, is used to exclude the characters within the square brackets from matching the target string.</p> <p>For example, the expression "[^0-9]" specifies that the target character must not be a numeral.</p>
\$ (Dollar symbol)	<p>The \$ is an anchor. The \$ matches the end of the string.</p> <p>For example, the expression "abc\$" matches the substring "abc" only if it is at the end of the string.</p>
(Vertical bar or pipe)	<p>The allows the character on either side of the vertical bar (or pipe) to match the target string.</p> <p>For example, the expression "a b" matches a as well as b.</p>
.(Period, Full stop or Dot)	<p>The . matches any character.</p>
*(Asterisk)	<p>The * indicates that the character to the left of the asterisk in the expression must match at least zero or more times.</p>
+(Plus symbol)	<p>The + indicates that the character to the left of the plus symbol in the expression must match at least once.</p>
?(Question mark)	<p>The ? indicates that the character to the left of the question mark must match at least zero or more than once.</p>
() (Parenthesis)	<p>The () enclosing a set of characters affects the order of pattern evaluation and also serves as a tagged expression that can be used when replacing the matched substring with another expression.</p>
[] (Square brackets)	<p>The [] enclosing a set of characters indicates that any of the enclosed characters might match the target character.</p>

For more information on regular expression, see the [Electronic Text Center \(http://etext.lib.virginia.edu/services/helpsheets/unix/regex.html\)](http://etext.lib.virginia.edu/services/helpsheets/unix/regex.html) or search the [Microsoft MSDN library. \(http://msdn.microsoft.com/en-us/library/default.aspx\)](http://msdn.microsoft.com/en-us/library/default.aspx)

NOTE: These reference sites provide you more detailed and comprehensive information on regular expressions, but only the expressions listed in the previous table are supported by Novell.

Command Reference

5

This section contains the following information:

- ♦ [Section 5.1, “Command Reference Conventions,” on page 53](#)
- ♦ [Section 5.2, “Commands,” on page 56](#)

5.1 Command Reference Conventions

This section consists of descriptions and examples of the commands that make up Novell® SecureLogin application definitions.

For a list of commands and corresponding page references, see [Chapter 1, “Quick Command Reference,” on page 11](#).

This section contains the following information:

- ♦ [Section 5.1.1, “Command Information,” on page 53](#)
- ♦ [Section 5.1.2, “Web Wizard Application Definition Conventions,” on page 54](#)
- ♦ [Section 5.1.3, “Integrating Novell Audit,” on page 55](#)
- ♦ [Section 5.1.4, “One-Time Passwords,” on page 56](#)

5.1.1 Command Information

This section contains the following information:

- ♦ [“Use With Values” on page 53](#)
- ♦ [“Type Values” on page 54](#)

Use With Values

Command	Description
Java	Use as part of a Java* application definition.
Startup	Use as part of a Startup application definition.
Terminal Launcher	Use as part of a terminal launcher application definition.
Advanced Web	Use as part of a manually created Web site or Internet application definition. Not compatible with the Web Wizard application definition language.
	NOTE: A predefined <i>Web</i> application and an <i>Advanced Web</i> application definition are the same.
Web Wizard	Use as part of application definitions created automatically by the Web Wizard. Web Wizard application definitions can be kept in their original XML format or converted to an ASCII script for advanced editing.

Command	Description
Windows	Use as part of a Window's application definition.

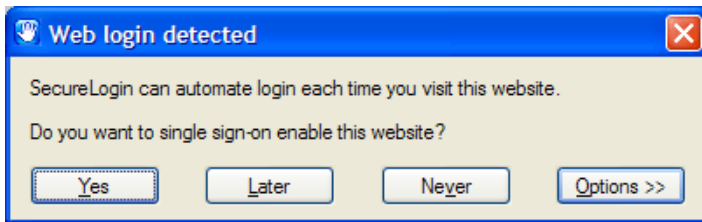
Type Values

Command	Description
Action	Performs an action. For example the <code>Type</code> command types information into a field.
Dialog specifiers	Defines dialog boxes. For example, the <code>Parent</code> and <code>Class</code> commands.
Flow control commands	Directs SecureLogin to a specific location in the application definition. For example, <code>Repeat</code> and <code>EndScript</code> commands.
Variable manipulators	Modifies variables, such as the <code>Add</code> and <code>Subtract</code> commands.

5.1.2 Web Wizard Application Definition Conventions

The SecureLogin advanced Web Wizard makes it easier for users to enable single sign-on Web sites and capture a user's Web-based login details. When the user accesses a Web page from the browser, SecureLogin automatically launches the Web Wizard.

Figure 5-1 *The Web Wizard*



The Web Wizard captures the user's login details and adds them to the user's Web application definitions.

When managing user's Web log in credentials, the *Definition* tab of the *Advanced Setting* page allows administrators to customize site and user credential details. Also available under the *Definitions* tab is an Advanced function that provides more functionality with their associated values and the option to convert the user's login credentials to an application definition.

For details on managing application definitions, see [Chapter 3, "Managing Application Definitions," on page 25](#).

- ◆ ["Site Matching" on page 55](#)
- ◆ ["Matching Form, Field, and Option" on page 55](#)
- ◆ ["Form, Field, and Option IDs" on page 55](#)

Site Matching

In SecureLogin version 6.0 and later, Web commands have been added to allow for much finer control of site matching. Detailed information of the loaded Web site can be matched and used to execute blocks of scripting commands.

The technique used to specify constraints upon a site match are similar to those constraints used in windows scripting.

Instead of `Dialog/EndDialog` commands, equivalent `Site/EndSite` commands have been created and can now be used.

Within these `Site` blocks, `Match` commands can be used to filter a given site. If one of the specified match commands fails to match, then the `Site` block fails to match as a whole. For more information, see [Section 5.2.76, "Site/Endsite," on page 151](#).

Matching Form, Field, and Option

When matching a specific form, field, or other match option, multiple items often match the selection criteria. In these cases, the first item on the Web site that matches is considered to be the match.

To access the other fields that also need to be matched, subsequent match commands can be added with the same selection criteria.

For example:

```
MatchField #1:1 -type "password"
MatchField #1:2 -type "password"
```

matches a site with two password fields. The first is given the ID '#1:1', the second is given the id '#1:2'

NOTE: ♦ Matched items might only be matched once.

- ♦ Each ID must be unique and not used previously.
-

Form, Field, and Option IDs

When matching a site, match methods are used to give specific fields, forms, and options their own unique ID.

After the site has been successfully matched, the given ID is used in input commands to specify particular items.

The actual IDs are denoted with a # followed by 1, 2, or 3 numbers each separated by a colon. For instance, "#1:3:2".

5.1.3 Integrating Novell Audit

Novell SecureLogin incorporates a Novell Audit integration for those enterprises that have Novell Audit as part of their infrastructure.

Novell Audit allows administrators to audit events from scripts to Novell Audit and Novell Sentinel™ servers in response to certain triggering events.

For details of the Novell Audit integration see [Section 5.2.4, “AuditEvent,” on page 64](#).

5.1.4 One-Time Passwords

The use of multiple passwords places a high maintenance overhead on large enterprises. Users are routinely required to use and manage multiple passwords, which can result in a significant cost, particularly with regard to calls to the help desk to reset forgotten passwords, or to ensure that all passwords are provisioned when a new user starts or are deleted when an existing user leaves the organization.

One of the main benefits of implementing one-time password systems is that it is impossible for a password to be captured on the wire and replayed to the server. This is particularly important if a system does not encrypt the password when it is sent to the server, as is the case with many legacy mainframe systems.

One-time passwords also offer advantages in terms of disaster recovery because the encryption key is used to generate the one-time password rarely changes. System restoration, which might be to a system version that is hours or many months old, can be achieved without consideration of restoring users' passwords or notifying staff of new passwords.

SecureLogin 6.1 also provides a secure, robust, and scalable infrastructure by integrating ActivCard* one time password authentication functionality. It provides administrators access to the application definition command `GenerateOTP`, which can be used to generate synchronous authentication and asynchronous authentication soft token support for smart card user authentication as well as hard token support for the Vasco* Digipass* token generator.

5.2 Commands

This section contains information on the following commands:

- ◆ [Section 5.2.1, “AAVerify,” on page 59](#)
- ◆ [Section 5.2.2, “ADD,” on page 62](#)
- ◆ [Section 5.2.3, “Attribute,” on page 63](#)
- ◆ [Section 5.2.4, “AuditEvent,” on page 64](#)
- ◆ [Section 5.2.5, “BeginSplashScreen/EndSplashScreen,” on page 64](#)
- ◆ [Section 5.2.6, “BooleanInput,” on page 65](#)
- ◆ [Section 5.2.7, “Break,” on page 66](#)
- ◆ [Section 5.2.8, “Call,” on page 68](#)
- ◆ [Section 5.2.9, “ChangePassword,” on page 69](#)
- ◆ [Section 5.2.10, “Class,” on page 70](#)
- ◆ [Section 5.2.11, “ClearPlat,” on page 71](#)
- ◆ [Section 5.2.12, “ClearSite,” on page 74](#)
- ◆ [Section 5.2.13, “Click,” on page 74](#)
- ◆ [Section 5.2.14, “ConvertTime,” on page 77](#)
- ◆ [Section 5.2.15, “Ctrl,” on page 77](#)
- ◆ [Section 5.2.16, “DebugPrint,” on page 78](#)

- ◆ Section 5.2.17, “Decrement,” on page 79
- ◆ Section 5.2.18, “Delay,” on page 80
- ◆ Section 5.2.19, “Dialog/EndDialog,” on page 81
- ◆ Section 5.2.20, “DisplayVariables,” on page 82
- ◆ Section 5.2.21, “Divide,” on page 83
- ◆ Section 5.2.22, “DumpPage,” on page 84
- ◆ Section 5.2.23, “EndScript,” on page 85
- ◆ Section 5.2.24, “Event/Event Specifiers,” on page 85
- ◆ Section 5.2.25, “FocusInput,” on page 86
- ◆ Section 5.2.26, “GenerateOTP,” on page 87
- ◆ Section 5.2.27, “GetCheckBoxState,” on page 89
- ◆ Section 5.2.28, “GetCommandLine,” on page 90
- ◆ Section 5.2.29, “GetEnv,” on page 91
- ◆ Section 5.2.30, “GetHandle,” on page 91
- ◆ Section 5.2.31, “GetIni,” on page 92
- ◆ Section 5.2.32, “GetMD5,” on page 92
- ◆ Section 5.2.33, “GetReg,” on page 93
- ◆ Section 5.2.34, “GetSessionName,” on page 94
- ◆ Section 5.2.35, “GetText,” on page 94
- ◆ Section 5.2.36, “GetURL,” on page 95
- ◆ Section 5.2.37, “GoToURL,” on page 96
- ◆ Section 5.2.38, “Highlight,” on page 96
- ◆ Section 5.2.39, “If/Else/EndIf,” on page 97
- ◆ Section 5.2.40, “Include,” on page 100
- ◆ Section 5.2.41, “Increment,” on page 101
- ◆ Section 5.2.42, “KillApp,” on page 102
- ◆ Section 5.2.43, “Local,” on page 103
- ◆ Section 5.2.44, “MatchDomain,” on page 104
- ◆ Section 5.2.45, “MatchField,” on page 105
- ◆ Section 5.2.46, “MatchForm,” on page 107
- ◆ Section 5.2.47, “MatchOption,” on page 109
- ◆ Section 5.2.48, “MatchReferer,” on page 110
- ◆ Section 5.2.49, “MatchTitle,” on page 111
- ◆ Section 5.2.50, “MatchURL,” on page 112
- ◆ Section 5.2.51, “MessageBox,” on page 113
- ◆ Section 5.2.52, “Multiply,” on page 115
- ◆ Section 5.2.53, “OnException/ClearException,” on page 116
- ◆ Section 5.2.54, “Parent/EndParent,” on page 121

- ◆ Section 5.2.55, “PickListAdd,” on page 123
- ◆ Section 5.2.56, “PickListDisplay,” on page 125
- ◆ Section 5.2.57, “PositionCharacter,” on page 126
- ◆ Section 5.2.58, “PressInput,” on page 127
- ◆ Section 5.2.59, “ReadText,” on page 128
- ◆ Section 5.2.60, “RegSplit,” on page 131
- ◆ Section 5.2.61, “ReLoadPlat,” on page 132
- ◆ Section 5.2.62, “Repeat/EndRepeat,” on page 134
- ◆ Section 5.2.63, “RestrictVariable,” on page 135
- ◆ Section 5.2.64, “Run,” on page 138
- ◆ Section 5.2.65, “Select,” on page 139
- ◆ Section 5.2.66, “SelectListBoxItem,” on page 140
- ◆ Section 5.2.67, “SelectOption,” on page 140
- ◆ Section 5.2.68, “SendKey,” on page 141
- ◆ Section 5.2.69, “Set,” on page 142
- ◆ Section 5.2.70, “SetCheckBox,” on page 143
- ◆ Section 5.2.71, “SetCursor,” on page 144
- ◆ Section 5.2.72, “SetFocus,” on page 145
- ◆ Section 5.2.73, “SetPlat,” on page 146
- ◆ Section 5.2.74, “SetPrompt,” on page 149
- ◆ Section 5.2.75, “-SiteDeparted,” on page 150
- ◆ Section 5.2.76, “Site/Endsite,” on page 151
- ◆ Section 5.2.77, “StrCat,” on page 153
- ◆ Section 5.2.78, “StrLength,” on page 154
- ◆ Section 5.2.79, “StrLower,” on page 155
- ◆ Section 5.2.80, “StrUpper,” on page 155
- ◆ Section 5.2.81, “Sub/EndSub,” on page 156
- ◆ Section 5.2.82, “Submit,” on page 157
- ◆ Section 5.2.83, “Subtract,” on page 158
- ◆ Section 5.2.84, “Tag/EndTag,” on page 160
- ◆ Section 5.2.85, “TextInput,” on page 160
- ◆ Section 5.2.86, “Title,” on page 161
- ◆ Section 5.2.87, “Type,” on page 162
- ◆ Section 5.2.88, “Using the Type Command to Send Keyboard Commands,” on page 166
- ◆ Section 5.2.89, “WaitForFocus,” on page 167
- ◆ Section 5.2.90, “WaitForText,” on page 169

5.2.1 AAVerify

Use With	Startup, Terminal Launcher, Web, or Windows
SecureLogin Version	3.5 to 6.1 SP1
Type	Action
Usage	<code>AAVerify [-Method <Defined method to use>] [-User <Username>] [-Tree <Tree name>] [?Result]</code>
Arguments	<p><code>-Method</code></p> <p>The name of the advanced authentication method you want to use. If a method is not specified <code>AAVerify</code> uses the method that was chosen during initial authentication to the directory.</p> <hr/> <p>NOTE: You can specify multiple methods.</p> <hr/> <p><code>-User</code></p> <p>The name of the user you want to use for the <code>AAVerify</code> command. If a method is, <code>AAVerify</code> reauthenticates the currently logged-in user.</p> <p><code>-Tree</code></p> <p>The name of the tree the user is in. You must use this with the <code>-User</code> argument.</p> <p><code>[?Result]</code></p> <p>A variable name (preferably a temporary variable) that receives the result of <code>AAVerify</code>. Set this variable to true for success or false for failure.</p> <p><code>?AAVerifyReturnCode</code></p> <p>A variable that is set with the error code that is generated from the <code>AAVerify</code> reauthentication process (if any).</p>

Description

Use `AAVerify` with SecureLogin Advanced Authentication or Novell NMAS to verify the user. It is typically used before the application username and password are retrieved and entered into the login box.

This provides application reauthentication using a strong login method. For example, a user might be forced to enter his or her smart card and PIN before the application login by using single sign-on, even though the application natively knows nothing about smart cards and PINs. If the verification succeeds, the `[?Result]` is set to true; otherwise, it is set to false. These additions are for Novell SecureLogin and NMAS.

NMAS Specific

If `AAVerify` is called with no arguments, then the currently logged-in user is reauthenticated by using the login method that he or she used for the current login.

AA Specific

When `AAVerify` is called in an AA environment, the method parameter must be present. The method must be one of the following:

- ◆ Any
- ◆ Biometric
- ◆ Smart card
- ◆ Token
- ◆ Password
- ◆ Passphrase
- ◆ Directory
- ◆ Password
- ◆ SecureID

You can specify more than one `-method` argument. In this case the user is allowed to reauthenticate with any of the specified methods. For example, you could use the command to request authentication by using a fingerprint device or a smart card.

NOTE: When the `AAVerify` command is added to an application definition, it only increases the security of the target application if it is not possible to alter the application definition. If the application definition could be modified or overridden, then the `AAVerify` command could be removed and there would be no additional security. For this reason it is imperative that application definition access be restricted through directory ACLs and SecureLogin's preferences, so that only a small, trusted group of administrators can modify, add and override application definitions.

Syntax examples

`AAVerify`

`AAVerify -Method "Enhanced Password" ?Result`

`AAVerify -Method "Enhanced Password"-User "BSmith" - Tree "Production" ?Result`

Example 1**Windows Application Definition**

This example detects the login dialog box, but before SecureLogin enters the user's credentials, it prompts the user to provide the Advanced Authentication credentials (smart card, PIN, biometric, and token)

```
# Log on Dialog Box
Dialog
Ctrl #32770
Title "Log on"
EndDialog
AAVerify -Method "Enhanced Password" ?Result
If ?Result Eq "True"
Type "$Username" #1001
Type "$Password" #1002
Click #1
Else
MessageBox "Authentication failed! Please verify
your smart card is inserted and your PIN is correct.
IT x453"
EndIf
```

Example 2**Windows Application Definition**

The following example shows the usage of the `OnExceptions` command as well as the `?AAVerifyReturnedCode`.

Refer to [Section 5.2.53, "OnException/ClearException," on page 116](#) for further details and examples of `OnException` usage.

```
#####  
# Login - Simple  
#####  
Dialog  
Title "Login - Simple"  
Class "#32770"  
Ctrl #1001  
Ctrl #1002  
Ctrl #1 "&Login"  
Ctrl #2 "Cancel"  
Ctrl #1027 "Username:"  
Ctrl #1028 "Password:"  
Ctrl #1009  
EndDialog  
OnException AAVerifyCancelled Call  
CancelSimpleLoginDialogCancelled  
OnException AAVerifyFailed Call  
CancelSimpleLoginDialogFailed  
AAVerify -method "smartcard"  
Type $Username #1001  
Type $Password #1002  
Click #1  
#  
# Cancel the Simple Login Window - AAVerify cancelled  
#  
Sub CancelSimpleLoginDialogCancelled  
Click #2  
EndScript  
EndSub  
#  
# Cancel the Simple Login Window - AAVerify failed  
#  
Sub CancelSimpleLoginDialogFailed  
Click #2  
MessageBox "Your re-authentication failed. Error "  
?AAVerifyReturnCode ". Login cancelled"  
EndScript  
EndSubg
```

5.2.2 ADD

Used With	Startup, Terminal Launcher, Web, or Windows
SecureLogin Version	3.0 to 6.1 SP1
Type	Variable Manipulator
Usage	Add <Variable1> <Variable2> [?Result]

Arguments	<p><Variable1></p> <p>The first argument; the number to which the second argument is added. This argument also contains the result of the addition equation if the optional [?Result] argument is not passed in. If used without the [?Result] argument, <Variable1> must be a SecureLogin variable. Otherwise, <Variable1> can be any numeric value.</p> <p><Variable2></p> <p>The second argument, the number added to the first argument in the equation. <Variable2> can be a SecureLogin variable or numeric value.</p> <p>[?Result]</p> <p>Optional. The sum, or the result of the equation.</p>
Description	<p>Adds one number to another. The numbers can be hard-coded into the application definition, or they can be variables. The result can be output to another variable, or to one of the original numbers.</p>
Syntax Examples	<pre>Add 1 2 ?Result Add ?LoginAttempts ?LoginFailures Add ?LoginAttempts ?LoginFailures ?Result Add ?LoginAttempts 3 Add ?LoginAttempts 3 ?Result</pre>
Example	<p>Windows Application Definition</p> <p>This example reads the values of Control IDs 103 and 104 into variables. From there they are added, and the result is typed into Control ID 1</p> <pre>ReadText #103 ?Number1 ReadText #104 ?Number2 Add ?Number1 ?Number2 ?Result Type ?Result #1</pre>

5.2.3 Attribute

Use With	Advanced Web Application Definition
SecureLogin Version	3.5 to 6.1 SP1
Type	Specifier
Usage	Attribute <Attribute Name> <Attribute Name>
Arguments	<p>< Attribute Name></p> <p>Name of the HTML Attribute to discover.</p> <p>< Attribute Value></p> <p>The value the above HTML Attribute must contain for the condition to be true.</p>

Description	Use the Attribute specifier in conjunction with the <code>Tag/EndTag</code> command to specify which HTML attributes and attribute values must exist for that particular HTML tag. For more information, see Section 5.2.84, "Tag/EndTag," on page 160 .
Example	This example finds the form that has an attribute of name with a value of login. <pre>Tag "Form" Attribute "Name" "Log on" EndTag</pre>

5.2.4 AuditEvent

Use With	Startup, Terminal Launcher, Web, or Windows application definitions for those enterprises that have Novell Audit as part of their infrastructure.
SecureLogin Version	6.0
Type	Specifier
Usage	<code>AuditEvent [<message>]</code>
Arguments	<code><message></code> The variable or text string passed to the Novell Audit server.
Description	Use <code>AuditEvent</code> to log <code>SecureLogin</code> events to the Novell Audit server. If the <code>Type</code> command is used with <code>ChangePassword</code> command to generate a <code>\$password</code> variable this triggers a log event to the Novell Audit server.
Example	If the Audit platform agent is not present on the workstation nothing is logged. <pre>AuditEvent "message"</pre> <p>The parameter "message" is the string passed to the Novell Audit server.</p> <pre>AuditEvent \$message</pre> <p>The parameter \$message is the variable passed to the Novell Audit server.</p>

5.2.5 BeginSplashScreen/EndSplashScreen

Use With	Terminal Launcher (Generic and Advanced Generic Only)
SecureLogin Version	3.0.4 to 6.1 SP1
Type	Action
Usage	<code>BeginSplashScreen</code> <code>EndSplashScreen</code>

Arguments	None
Description	Use to display a splash screen across the whole Terminal Emulator window. This is used to mask any flashing or other effects that are produced by SecureLogin scraping the screen for text. A <code>Delay</code> command at the start of the application definition ensures that the emulator window is in place before the splash screen is displayed.
Example	<p>Terminal Launcher Application Definition</p> <p>This example launches the emulator and SecureLogin waits two seconds for it to connect. The splash screen is displayed to cover the flashing, the login is detected, the username is entered, then the splash screen disappears.</p> <pre>Delay 2000 BeginSplashScreen WaitForText "login:" Type \$Username EndSplashScreen Type @E</pre>

5.2.6 BooleanInput

Use With	Advanced application definitions created by using the Web Wizard.
SecureLogin Version	3.6.1.0 to 6.1 SP1
Type	Action
Usage	<code>BooleanInput #FormID:FieldID -check "check"</code>
Arguments	<p><code>#FormID</code></p> <p>The ID to be given to the matched form. The ID must be a static unsigned integer.</p> <p><code>#FieldID</code></p> <p>The ID to be given to the matched field. The ID must be a static unsigned integer.</p> <p><code>-check "check"</code></p> <p>"check" is a Boolean value indicating a set or unset state for the specified field.</p>
Description	Used inside a site block to set the state of a Boolean field (either a checkbox or radio button).

Example

In this example, the value of field #1:3 is being checked by the application definition.

```
# === Login Application Definition #2 ==
# === Google Initial Login ===
#####
Site Login -userid "Google Log On" -initial
MatchDoimain "www.google.com"
MatchField #1:1 -name "Email" -type "text"
MatchField #1:2 -name "Passwd" -type "password"
MatchField #1:3 -name "Cookie" -type "check"
EndSite
SetPrompt "Enter your user credentials"
TextInput #1:1 -value "$Username"
TextInput #1:2 -value "$Password"
FocusInput#1:2 -focus "true"
BooleanInput #1:3 -check "false"
PressInput
Endscript
```

5.2.7 Break

Use With	Startup, Terminal Launcher, Web, and Windows
SecureLogin Version	3.5 to 6.1 SP1
Type	Action
Usage	Break
Arguments	None
Description	Use Break within the Repeat/EndRepeat commands to break out of a repeat loop.
Example 1	Windows Application Definition This example reads the screen and the content is searched for the word log on. If log on is found, the Repeat loop is broken and the application definition continues. If log on is not found, the application definition checks again. Dialog Class #32770 Title "Log on" EndDialog Repeat ReadText #301 "?Text" If ?Text Eq "Log on" Break EndIf Delay 100 EndRepeat

Example 2**Terminal Application Definition**

This example reads the terminal emulator screen and the content is searched for a successful login (in this case the application main menu appears). After the user has logged in, the Repeat loop is broken and the application definition continues. If the login is not successful, the application definition checks again. Terminal Emulators uses repeat loops for error handling and to break out of the loop as appropriate.

```
# Initial System Login
WaitForText "ogin:"
Type $Username
Type @E
WaitForText "assword:"
>Type $Password
Type @E
Delay 500
# Repeat loop for error handling
Repeat
Check to see if password has expired
If -Text "EMS: The password has expired."
ChangePassword $Password
Type $Password
Type @E
Type $Password
Type @E
EndIf
#User has an invalid Username and / or Password
stored.
If -Text "Log on Failed" DisplayVariables "The
username and / or password stored by SecureLogin is
invalid. Please verify your credentials and try
again. IT x453."
Type $Username
Type @E
Delay 500
WaitForText "assword:"
Type $Password
Type @E
Delay 500
EndIf#

Account is locked for some reason, possibly inactive.
If -Text "Account Locked" MessageBox "Your account
has been locked, possibly due to inactivity for 40
days.
Please contact the administrator on x453." EndIf#
Main
Menu, user has logged on #successfully. If
-Text "Application Selection" Break
EndIf
Delay100
EndRepeat
```

5.2.8 Call

Use With	Startup, Terminal Launcher, Web, or Windows
SecureLogin Version	3.5 to 6.1 SP1
Type	Flow control
Usage	Call <SubRoutine>
Arguments	<SubRoutine> The name of the subroutine called. This name must be identical to the name specified in the <code>Sub</code> command.
Description	<p>Use the <code>Call</code> command to call and run a subroutine. When a subroutine is called, the application definition begins executing from the first line of the subroutine.</p> <p>When the subroutine completes, the application definition resumes executing from the command immediately following the <code>Call</code> command.</p>
Example	<p>Terminal Application Definition</p> <p>This example looks for the word <code>Username</code>. If it is found on the screen, the subroutine <code>login</code> is launched. If <code>Wrong Password</code> is found, the subroutine <code>WrongPassword</code> is launched.</p> <p>Subroutines are useful when you would otherwise need to repeat the same lines of the application definition over again.</p> <pre>Repeat If -Text "Username" Call "Login" EndIf If -Text "Wrong Password" Call "WrongPassword" EndIf Delay 100 EndRepeat ===Login Subroutine=== Sub Login Type \$Username Type @E Type \$Password Type @E EndSub ===Wrong Password Subroutine=== Sub WrongPassword DisplayVariables "The password entered is incorrect. Please verify your password and click OK to retry log on. IT x453." \$Password Call Login EndSub</pre>

5.2.9 ChangePassword

Use With	Startup, Terminal Launcher, Web, and Windows
SecureLogin Version	3.5 to 6.1 SP1
Type	Action
Usage	ChangePassword <Variable> [<Text>] "Random"
Arguments	<Variable> A normal or runtime variable in which the password is stored. [Text] The text you want displayed in the Change Password dialog box. [Random] Random invokes the random password generator.
Description	Use <code>ChangePassword</code> to change a single variable. It is also used scenarios where password expiry is an issue. Set the <Variable> to the new password. The flag for this command is Random. If Random is: <ul style="list-style-type: none">◆ Set, the new password is generated automatically in compliance with the variable's password policy.◆ Not set, a dialog box prompts the user to enter a new password. The new password is tried against any variable password policies that are in place. See also Section 5.2.63, "RestrictVariable," on page 135.
Syntax Examples	<pre>ChangePassword \$NewPassword ChangePassword ?NewPassword "Please enter a new password" ChangePassword ?NewPassword Random</pre>

Example**Windows Application Definition**

This example detects the change password event. The application requires the current username and password, and then the new password and confirmation of the new password. The application definition creates a backup of the old password in case the password change fails (which is detected by the message that is displayed), and then generates and enters a new password.

```
# Change Password Dialog
BoxDialog
Class #32770
Title "Change Password"
EndDialog
Set $PasswordBackup $Password
Type $Password #1015
ChangePassword $Password Random
Type $Password #1005
Type $Password #1006
Click #1#
Change Password Failed Dialog Box
Dialog
Class #32770
Title "Change Password Failed"
EndDialog
# Set the password back as the password change
failedSet
$Password $PasswordBackup
MessageBox "The change password process failed.
Please
retry the password change at your next log on. IT
x453."
```

5.2.10 Class

Use With	Startup, Windows
SecureLogin Version	3.5 to 6.1 SP1
Type	Dialog Specifier
Usage	Class <Window-Class>
Arguments	<Window-Class>

A string specifying the window class that this statement matches.

Description When a window is created, it is based on a template known as a window class. The `Class` command 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 application definition continues to the next line.
- ◆ Does not match the `<Window-Class>` argument, execution continues at the next dialog statement.

NOTE: Use the Window Finder tool to determine the window class.

Example 1 Windows Application Definition

This example checks the dialog box generated by the application to determine if the window class is #32770. If true and its title is log on, that section of the application definition executes. If false, the application definition checks the next dialog block.

```
# Log on Dialog Box
Dialog
  Class "#32770"
  Title "Log on"
EndDialog
  Type $Username #1001
  Type $Password #1002
Click #1
```

Example 2 Java Application Definition

You can now use the dialog statements in Java definitions to specify the class of the window, similar to Windows definitions.

```
Dialog
Title "Login"
Class "PswFrame"
EndDialog
Messagebox test
```

The Class information is listed in the definition created by the Add Java Application wizard.

5.2.11 ClearPlat

For each dialog block in an application definition, the chosen User ID is reset and you must select it again. Select it again by using a `SetPlat` command or by having the user select again from a list.

When an application first presents a login screen, `SecureLogin` directs the user to select an appropriate User ID from a list. `SecureLogin` enters the selected User ID's credentials into the application and submits them.

If the login fails because incorrect credentials, `SecureLogin` prompts the user to change the credentials. `SecureLogin` does not retain User ID details and prompts the user to reenter them. However, this could result in changing the wrong credentials if the user selects the incorrect User ID.

To resolve this issue, use the `SetPlat`, `ReLoadPlat`, and `ClearPlat` commands. `ReLoadPlat` sets the current User ID to the one which was last chosen (for the given application), or leaves the User ID unset if a User ID has not been selected previously. `ClearPlat` resets the last chosen User ID.

See also [Section 5.2.61, “ReLoadPlat,” on page 132](#) and [Section 5.2.11, “ClearPlat,” on page 71](#).

Use With	Startup, Terminal Launcher, Web, or Windows
SecureLogin Version	V.3.6.0 to 6.1 SP1
Type	Action
Usage	<p>There are three main places where code needs to be added to use the <code>ClearPlat</code> command:</p> <p>Application Startup</p> <p>When an application first starts up, use <code>ClearPlat</code> to clear the previously chosen platform. (Do this in a Windows application by adding an extra dialog statement for the main window).</p> <p>Change Credentials Canceled</p> <p>Call <code>ClearPlat</code> if the user decides not to modify the chosen platform's credentials, thus giving him or her a chance to choose a different platform next time.</p> <p>Successful Login</p> <p>Call <code>ClearPlat</code> to allow the user to login again with a different platform at a later stage</p>
Arguments	None
Description	Use to reset the last chosen platform, causing subsequent calls to <code>ReLoadPlat</code> to do nothing.

Example

```
Windows Application Definition
#== BeginSection: Application startup ====
Dialog
Class "#32770"
Title "Password Test Application"
EndDialog
ClearPlat
# == EndSection: Application startup=====
# ===== BeginSection: Log on =====
Dialog
Class "#32770"
Ctrl #1001
Title "Log on"
EndDialog
ReloadPlat
SetPrompt "Username =====>"
Type $Username #1001
SetPrompt "Password =====>"
Type $Password #1002
SetPrompt "Domain =====>"
>Type $Domain #1003
Click #1
# ===== EndSection: Log on =====
## =====BeginSection: Log on Successful =====
Dialog
Class "#32770"
Title "Log on Successful"
EndDialog
ClearPlat
```

Example (Cont.)

```
Click #2
# ===== EndSection: Log on Successful =====
# ===== BeginSection: Log on Failure =====
Dialog
Class "#32770"
Title "Log on Failure"
EndDialog
Click #2
ReloadPlat
OnException ChangePasswordCancelled Call
ChangeCancelled
ChangePassword $password
ClearException ChangePasswordCancelled
Type -raw \Alt+F
Type -raw L
# ===== EndSection: Log on Failure =====
# ===== BeginSection: Change Credentials Cancelled
=====
Sub ChangeCancelled
ClearPlat
EndScript
EndSub
# ===== EndSection: Change Credentials
Canceled ===  


---


```

5.2.12 ClearSite

Use With	Startup, Terminal Launcher, Web, Windows, and Advanced application definitions created by using the Web Wizard.
SecureLogin Version	3.6.1.0 to 6.1 SP1
Type	Action
Usage	ClearSite "SiteName"
Arguments	"SiteName" The name of the site to clear.
Description	Used inside a site block to clear the matched status for a given site. This allows -initial sites to match again and causes -recent and -subsequent sites to fail to match. The <code>ClearSite</code> command needs to have the complete URL specified in the line before the <code>ClearSite</code> command.
Example 1	In this example, the user is redirected to the main Google portal and any previous user information is cleared. <pre>GotoURL "http://www.google.com" ClearSite Login</pre>
Example 2	In this example, the <code>ClearSite</code> command is used with as part of conditional statement and if a particular condition is true the user information is cleared. <pre>MessageBox "Would you like to login again?" -yesno ?Continue If ?Continue eq "Yes" TextInput #1:1 -value "\$Username" TextInput #1:2 -value "\$Password" FocusInput #1:2 -focus "true" BooleanInput #1:3 -check "false" PressInput Else ClearSite Login EndIf</pre>

5.2.13 Click

Use With	Java, Web, Windows
SecureLogin Version	3.5 to 6.1 SP1
Type	Action
Windows Usage	Usage One:Click <#Ctrl-ID> [-Raw] [-Right]Usage Two:Click <# Ctrl-ID > [-Raw [-x < X Co-ordinate > -y <Y Co-ordinate >]]
Web Usage	Click <#Number>

Arguments

<#Ctrl-ID>

The ID number of the control to be pressed.

[-Raw]

-Raw eliminates the mouse and sends a direct click.

[-Right]

-Right, used only with the -Raw flag, sends a right mouse click.

<X Co-ordinate>

X represents the horizontal coordinate relative to the client area of the application (not the screen).

<Y Co-ordinate>

Y represents the vertical coordinate relative to the client area of the application (not the screen).

<#Number>

The pound/hash symbol followed by the sequential number/control ID of the button to be pressed.

Web specific

The number of the button is determined by the Web page layout. See [Section 5.2.22, "DumpPage," on page 84](#).

Windows specific

This is the control ID. Use the Windows Finder tool to discover the control ID.

Java specific

The index to use is put in an example application definition created by the Java wizard.

Description	<p>When used with windows applications, the <code>Click</code> command sends a click instruction to the specified <code><#Ctrl-ID></code>.</p>
	<hr/> <p>NOTE: If the button to be clicked does not have a control ID, the <code>Type "\N"</code> command often clicks the default button in a Windows application.</p> <hr/>
	<p>You can set the <code>-Raw</code> flag if the button or control does not respond to the <code>Click</code> command. The <code>-Raw</code> flag causes SecureLogin to emulate the mouse and send a direct click message to the control. Using the <code>-Right</code> flag with the <code>-Raw</code> flag sends a right-click to the control.</p>
	<p>Setting the <code><#Ctrl-ID></code> to 0 (zero) sends the click instruction to the window on which the application definition is running.</p>
	<p>If <code>-Raw</code> is specified, then you can set the X coordinate and the Y coordinates. These coordinates are relative to the client area of the application, not the screen.</p>
	<p>When used with Web application definitions, the <code>Click</code> command takes a single argument, which is the sequential number on the page of the button to be pressed. <code>Click #3</code> clicks the third button on the page. Keep in mind that due to Web page layout and design, the sequential order of the buttons might not be obvious, and that you might have to use the <code>DumpPage</code> (see Section 5.2.22, "DumpPage," on page 84) command to discover the field layout.</p>
Syntax Examples	<pre>Click #1 Click #1 -Raw -Right Click -X 12 -Y 24</pre>
Example 1	<p>Windows Application Definition</p>
	<p>This example detects the login dialog box, the username and password are entered, and button number 1 (in this case the login button) is clicked.</p>
	<pre># Log on Dialog Box Dialog Class #32770 Title "Log on" EndDialog Type \$Username #1001 Type \$Password #1002 Click #1</pre>
Example 2	<p>Web Application Definition</p>
	<p>This example enters the username and password, and then the login button is clicked.</p>
	<pre>Type \$Username Type \$Password Password Click #1</pre> <hr/>

Example 3**Windows Application Definition**

This example uses the Java application, so there is no Control ID. Instead, the `Click` command is told to click a particular place on the window.

```
# Log on Dialog Box
Dialog
  Class #32770
  Title "Log on"
End Dialog
Type $Username
Type $Password
Click -X 12 -Y 24
```

5.2.14 ConvertTime

Use With	Startup, Terminal Launcher, Web, and Windows
SecureLogin VersionSecureLogin Version	3.0.4 to 6.1 SP1
Type	Variable Manipulator
Usage	<code>ConvertTime <time> <String Time></code>
Arguments	<code><String Time></code> The output variable.
Description	Convert a numeric time value, for example, <code>?CurrTime(system)</code> , into a legible format and stores it in <code><String Time></code> .
Example	Windows Application Definition This example converts the time to a readable format, and displays it in a dialog box. <pre># Log on Dialog Box Dialog Class #32770 Title "Log on" End Dialog ConvertTime ?CurrTime(system) ?Time MessageBox ?Time</pre>

5.2.15 Ctrl

Use With	Startup, Windows, Java
SecureLogin Version	3.5 to 6.1 SP1
Type	Dialog Specifier
Usage	<code>Ctrl <#Ctrl-ID> [<Regular Expression>]</code>

Arguments	<p><#Ctrl-ID></p> <p>The ID number of the control to check.</p> <p>[<RegEx>]</p> <p>The regular expression.</p>
Description	<p>Use the <code>Ctrl</code> command to determine 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.</p> <hr/> <p>NOTE: Third-party software control ID numbers might not be consistent from one version to the next. Use the Window Finder tool to determine the control ID.</p> <hr/> <p>Using the [<RegEx>] argument adds a further check that allows the application definition to skip to the next command. If the text on the specified <#Ctrl-ID> does not conform to the [<RegEx>], the application definition skips to the next dialog statement as though the <#Ctrl-ID> did not exist.</p>
Syntax Examples	<pre>Ctrl #1 Ctrl #1 "OK"</pre>
Example	<p>Windows Application Definition</p> <p>This example tests the dialog box to see if it contains the correct Control IDs with the correct values. If any of the Control IDs are missing, or the text does not match, the application definition passes on to the next dialog block.</p> <pre># Log on Dialog Box Dialog Ctrl #1 "OK" Ctrl #2 "Cancel" Ctrl #3 "Help" Title "Log on" EndDialog Type \$Username Type "\T" Click #1</pre>

5.2.16 DebugPrint

Use With	All
SecureLogin Version	6.0
Type	Action
Usage	DebugPrint <data>
Arguments	<p><data> The text displayed to the user.</p> <p><Data> can be several strings, variables, or a combination of both.</p>

Description	Use the <code>DebugPrint</code> command to display the text specified in the <code><Data></code> variable on a Debug console. The command can take any number of text arguments, including variables, (for example <code>DebugPrint "The user " \$Username " has just been logged in to the system"</code>).
Syntax Examples	<pre>DebugPrint "Caught the login dialog" DebugPrint "Setting platform to " ?Platform</pre>
Example	<p>Windows Application Definition</p> <p>This example displays the text specified in the <code>?ServerName</code> variable on the Debug console.</p> <pre># Login Dialog # Dialog Class "#32770" Title "Log on" EndDialog ReadText #1003 ?ServerText RegSplit "Server: (.*)" ?ServerText ?ServerName DebugPrint "Setting the platform to " ?ServerName SetPlat ?ServerName Type \$Username #1001 Type \$Password #1002 Click #1</pre>

5.2.17 Decrement

Use With	All
SecureLogin Version	3.5 to 6.1 SP1
Type	Variable Manipulator
Usage	<code>Decrement <Variable></code>
Arguments	<p><code><Variable></code></p> <p>The name of the variable to decrease in value.</p>
Description	<p>Use the <code>Decrement</code> command to subtract from a specified variable. For example, you can use <code>decrement</code> to count the number of passes a particular application definition has made.</p> <p>After the number of instances is equal to the specified number, you can instruct the application definition to run another task or end the application definition. This is useful when configuring an application whose login panel is similar to other windows within the application, or to easily control the number of attempts a user can have to access an application.</p> <p>Also see Section 5.2.41, "Increment," on page 101</p>
Syntax examples	<code>Decrement ?RunCount</code>

Example**Windows Application Definition**

Each time the application definition is run, a variable is incremented. This example counts the number of times the dialog box is displayed. If the dialog box is displayed more than three times, the application is closed. If the login is successful, the count is reset.

```
#Log on Dialog Box
Dialog
Class #32770
Title "Log on"
EndDialog
Decrement ?RunCount
If ?RunCount Gt "3"
MessageBox "Log on has been attempted too many times.
The
application will be closed."
KillApp "app.exe"
Else
Type $Username #1001
Type $Password #1002
Click #1
EndIf
# Log on Successful Message
Dialog
Ctrl #1
Title "Log on Successful"
EndDialog
Set ?RunCount "0"
```

5.2.18 Delay

Use with	All
SecureLogin Version	3.5 to 6.1 SP1
Type	Action
Usage	Delay <Time Period>
Arguments	<Time Period>
	A period of time, expressed in milliseconds (1/1000 of a second), during which the application definition execution is paused.
Description	Use the <code>Delay</code> command to delay the execution of the application definition for the time specified in the <Time Period> argument. The time specified in the <Time Period> argument is noted in milliseconds (for example, <code>Delay 5000</code> creates a five second pause). You can use the <code>Delay</code> command to accommodate an introduction screen or another custom feature.

Example**Windows Application Definition**

This example detects the login box, but the application definition waits half a second before acting upon it to make sure that the box is complete.

```
# Log on Dialog Box
Dialog
  Class #32770
  Title "Log on"
EndDialog

Delay 500
Type $Username #1001
Type $Password #1002
Click #1
```

5.2.19 Dialog/EndDialog

Use With	Java, Windows
----------	---------------

SecureLogin Version	3.5 to 6.1 SP1
---------------------	----------------

Type	Dialog Specifier
------	------------------

Usage	DialogEndDialog
-------	-----------------

Arguments	None
-----------	------

Description	<p>Use the <code>Dialog/EndDialog</code> command to identify the beginning and end of a dialog specification block. You can use these commands to construct a dialog specification block, which consists of a series of dialog specification statements (for example <code>Ctrl</code>, <code>Title</code>, and so on).</p>
-------------	---

When a dialog block is executed, each of the dialog specification statements is executed in sequence. If any statement within the dialog block is not found, the entire dialog block is considered false, and then application definition execution proceeds to the next dialog block, if any. You need to specify as much information in the dialog block to make the dialog box (for example, `Login`, `Change Password`, and so on) unique.

The portion of the application definition that follows the `EndDialog` command is called the application definition body. Another dialog block, or the end of the application definition, terminates the application definition body.

Example**Windows Application Definition**

This example tests the dialog box in order to determine its identity. If it is determined to be the login box, the application definition parses the `Type` and `Click` commands to complete the login process.

```
# Log on Dialog Box
Dialog
  Ctrl #1 "OK"
  Title "Log on"
  Parent
  Title "Application 1"
  EndParent
EndDialog

Type $Username #1001
Type $Password #1002
Click #1
```

5.2.20 DisplayVariables

Use With	All
SecureLogin Version	3.5 to 6.1 SP1
Type	Action
Usage	DisplayVariables [<User Prompt>] [<Variable> [<variable>] ...]
Arguments	[<User Prompt>] Optional, customized text displayed in the Enter SecureLogin Variables dialog box. [<Variables>] The name of the variables for which you want the user prompted. If not specified, SecureLogin prompts for all variables that are used by the application definition.

Description	<p>Use the <code>DisplayVariables</code> command to display a dialog box that lists the user's stored variables (for example, <code>\$Username</code> and <code>\$Password</code>) for the current application.</p> <p>About Editing Variables</p> <p>The user can edit the variables from this dialog box. For example, if the log in is unsuccessful because of an incorrect username or password, the <code>DisplayVariables</code> command prompts the user to edit the stored username or password values. The login process proceeds as normal from that point. You can specify a particular variable to display.</p> <p>If the <code><variables></code> parameter is specified, <code>DisplayVariables</code> prompts only for the variables specified. Enter the replacement text in quotation marks after the <code>DisplayVariables</code> command. This replaces the default prompt text in the <i>Enter SecureLogin Variables</i> dialog box.</p> <p>If there are no variables stored for the user, the first time <code>SecureLogin</code> attempts to single sign-on to the application, the prompt is not customized.</p> <p>After there are variables stored for the user, the prompt is customized when the application definition is run. The <code>SetPrompt</code> command can also be used to customize the prompt text in the dialog box.</p> <hr/> <p>NOTE: You can use the <code>OnException EnterVariablesCancelled</code> command to prevent a user from canceling the <code>DisplayVariables</code> prompt.</p>
-------------	--

Syntax Examples	<pre>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</pre>
-----------------	---

Example	<p>Windows Application Definition</p> <p>This example detects the Wrong Password dialog, and <code>SecureLogin</code> prompts the user to enter a new username and password. After they are specified, <code>SecureLogin</code> enters them into the dialog box, and the user clicks OK.</p> <pre># Wrong Password Dialog Box Dialog Class #32770 Title "Wrong Password" EndDialog DisplayVariables "Enter a new username and password"?\$Username \$Password Type \$Username #1001 Type \$Password #1002 Click #1</pre>
---------	---

5.2.21 Divide

Use With	Startup, Terminal Launcher, Web, or Windows
----------	---

SecureLogin Version	3.0 to 6.1 SP1
Type	Variable Manipulator
Usage	Divide <Variable1> <Variable2> [?Result]
Arguments	<p><Variable1></p> <p>The dividend, the first argument, the number that is divided by the second argument. Also, this argument contains the result if the optional [?Result] argument is not passed in. If used without the [?Result] argument, <Variable1> must be a SecureLogin variable, either?Variable1 or \$Variable1. Otherwise <Variable1> can be any numeric value.</p> <p><Variable2></p> <p>The divisor, the second argument, the number by which the first argument is divided. <Variable2> can be a SecureLogin variable or a numeric value.</p> <p>[?Result]Optional. The quotient, or the result of the equation.</p>
Description	<p>Use to divide one number by another. The numbers can be hard-coded into the application definition, or they can be variables. The result can be output to another variable, or to one of the original numbers.</p> <hr/> <p>NOTE: This is an integer arithmetic that is 5/2, not 2.5.</p> <hr/>
Syntax Examples	<pre>Divide "1" "2" ?Result Divide ?LoginAttempts ?LoginFailures Divide ?LoginAttempts ?LoginFailures ?Result Divide ?LoginAttempts "3" Divide ?LoginAttempts "3" ?Result</pre>
Example	<p>Windows Application Definition</p> <p>This example reads the values of Control IDs 103 and 104 into variables. From there they are divided, and typed into Control ID 1.</p> <pre>ReadText #103 ?Number1 ReadText #104 ?Number2 Divide ?Number1 ?Number2 ?Result Type ?Result #1</pre>

5.2.22 DumpPage

Use With	Advanced Web Application Definition
SecureLogin Version	3.5 to 6.1 SP1
Type	Action
Usage	DumpPage <Variable>
Arguments	<p><Variable></p> <p>The string variable to receive the page information.</p>
Description	<p>Use the DumpPage command to provide information about the current Web page. Use for debugging Web page application definitions.</p>

Example	DumpPage ?dump MessageBox ?dump
---------	------------------------------------

5.2.23 EndScript

Use With	Startup, Terminal Launcher, Web, or Windows
SecureLogin Version	3.5 to 6.1 SP1
Type	Action
Usage	EndScript
Arguments	None
Description	Use the <code>EndScript</code> command to immediately terminate execution of the application definition.
Example	Windows Application Definition

This example detects the login box, and SecureLogin enters the username and password, and the user clicks *OK*. If the Incorrect Password message is detected, SecureLogin displays a message that the password was incorrect, and terminates the application definition.

```
Dialog
  Title "Log on Failure"
  Ctrl #1
EndDialog

ReadText #65535 ?ErrorMsg
If "Incorrect Password" -In ?ErrorMsg MessageBox "You
have entered an incorrect password"
  EndScript
EndIf
```

5.2.24 Event/Event Specifiers

Use With	Windows
SecureLogin Version	3.5 to 6.1 SP1
Type	Dialog Specifier
Usage	Event <Event Specifier>

Arguments	<p><Event Specifier></p> <p>The application event to monitor. This corresponds to a Windows event, which usually begins with WM_.</p> <p>For example, WM_COPYDATA, WM_GETOBJECT, WM_GETTEXT</p> <p>For detailed information on Windows events, see the Microsoft MSDN Web site (http://msdn.microsoft.com).</p> <p>Microsoft Spy++, or similar Windows message spy tools, are also useful for trapping event names in specific windows. Information regarding Spy ++ is also available on the MSDN Web site.</p>
Description	<p>Application definitions generally execute at the point when an application window is created. This corresponds to the WM_CREATE message that is received from an application window at startup. By adding the Event specifier to a dialog block, you can override this behavior, so that an application definition only executes when (and only when) the specified message is generated. If no Event specifier is given, it is equivalent to Event WM_CREATE.</p> <p>You can only apply the Event specifier within a Dialog and EndDialog statement block. Only one Event can be specified per Dialog block. If there is a requirement to monitor for multiple events, each must be specified within its own dialog block.</p> <p>For more information, see the MSDN Web site or other documentation on the Win32 messaging system.</p>
Syntax Examples	<pre>Dialog Class "someclass" Event WM_ACTIVATE EndDialog MessageBox "Caught the WM_ACTIVATE message"</pre>

5.2.25 FocusInput

Use With	Startup, Terminal Launcher, Web, Windows, and Advanced application definitions created by using the Web Wizard.
SecureLogin Version	3.6.1.0 to 6.1 SP1
Type	Action
Usage	FocusInput #FormID:FieldID [-focus "focus"]

Arguments	<p>#FormID</p> <p>The ID to be given to the matched form. The ID must be a static unsigned integer.</p> <p>#FieldID</p> <p>The ID to be given to the matched field. The ID must be a static unsigned integer.</p> <p>-focus "focus"</p> <p>Focuses the input field based upon the Boolean value of "focus". The Boolean value can be either true or false.</p>
Description	Used to focus on an input field based upon the Boolean value of "focus".
Example	<p>In this example, the value of field #1:2 is being checked by the application definition.</p> <pre># === Login Application Definition #2 == # === Google Initial Login ==== #===== Site Login -userid "Google Log On" -initial MatchDoimain "www.google.com" MatchField #1:1 -name "Email" -type "text" MatchField #1:2 -name "Passwd" -type "password" MatchField #1:3 -name "Cookie" -type "check" EndSite SetPrompt "Enter your user credentials" TextInput #1:1 -value "\$Username" TextInput #1:2 -value "\$Password" FocusInput #1:2 -focus "true" BooleanInput #1:3 -check "false" PressInput Endscript</pre>

5.2.26 GenerateOTP

Use With	Startup, Terminal Launcher, Web and Windows
SecureLogin Version	3.5 to 6.1 SP1
Type	Action
Usage	GenerateOTP -mode <string>-challenge <string>

Arguments	<p><result></p> <p>A variable that receives the value of the one time password that is generated.</p> <p>-mode</p> <p>Specifies the type one-time password that is dynamically generated. The default value for mode is set to "soft" for the Vasco soft token. Setting this to "AISC-SKI" makes SecureLogin use the algorithm to generate an one-time password based on the user's smart card.</p> <p>-challenge</p> <p>When the one-time password generated is based on a challenge/response or asynchronous mode, the challenge needs to be passed to the <code>GenerateOTP</code> command as an argument, normally by means of a script that reads the challenge from the screen.</p>
Description	<p>A one-time password is an authentication method specifically designed to avoid the security exposures inherit with traditional fixed and static password usage.</p> <p>One-time passwords rely upon a predefined relationship between the user and the authenticating server. The encryption key is shared between the user's token generator and the server, with each performing the pseudo-random code calculation at user login. If the codes match, the user is authenticated.</p> <p><code>GenerateOTP</code> was an undocumented command initially developed in SecureLogin Version 3.5.x to meet a specific client requirement and was for use with the Vasco Digipass hard token generator.</p> <p>In SecureLogin Version 6, the <code>GenerateOTP</code> command was enhanced to incorporate one-time password soft token generation functionality embedded in smart card functionality.</p> <p>Soft tokens can be generated in synchronous and asynchronous mode, which now allows soft tokens to be loaded onto mobile devices such PDAs and even be sent to cell phones as SMS text messages.</p> <p>Synchronous Mode: Synchronous authentication or patented time-plus event authentication replaces static alphabetic and numeric passwords with a pseudo-random code that is dynamically generated at configured time intervals generally around each 60 seconds. The pseudo random code is based on a shared encryption key and the current time.</p> <p>Asynchronous Mode: Asynchronous authentication or challenge response authorization replaces static alphabetic and numeric passwords with a pseudo random code that is dynamically generated based on a shared encryption key, the current time and a challenge/response combination. In Asynchronous mode, the challenge needs to be passed to the <code>GenerateOTP</code> command as an argument.</p> <p>The application definition synchronous example shows a typical command structure to enable one-time password for use with a Vasco Digipass hard token generator.</p> <p>The application definition asynchronous example shows a typical command structure to enable one-time password for use with Smart Card technology.</p>

Example

In SecureLogin Version 6, the `GenerateOTP` command has been enhanced to integrate with smart cards.

In Synchronous mode, the `GenerateOTP` command requires the administrator to pass the `-mode` variable, `AISC-SKI` to the command.

In this instance, `AISC-SKI` is the smart card and `SKI` is the name of the applet used on the smart card.

An example application definition enabling synchronous one-time password encryption key distribution for use with smart cards is as follows:

```
Dialog
  Title "Test App"
EndDialog
  GenerateOTP -mode "AISC-SKI" ?OtpResult
  Type ?OtpResult #14
```

In Asynchronous mode the challenge needs to be passed to the `GenerateOTP` command as an argument. This requires a script that reads the challenge variable from the screen.

An example application definition enabling asynchronous one-time password encryption key distribution for use with smart cards is as follows:

```
Dialog
  Title "Test App"
EndDialog
  ReadText #12 ?tmp
  GenerateOTP -mode "AISC-SKI" -challenge ?tmp ?Otp
  Type ?Otp #14
```

It is assumed that a call without a challenge passed in is synchronous.

The `-mode` parameter, instead of being passed in via the script, can also be created as a single sign-on variable in the script platform.

If the `-mode` parameter is not passed in as a parameter to the `GenerateOTP` command SecureLogin checks for a variable named `mode` before assuming the default which is to generate a Vasco Token. Values passed into the command via the script override values defined as variables. This is for future integration with SecureLogin For Mobiles.

NOTE: It is assumed that the `acomx.dll` is present on the machine and in the path. If not, then additional code might be required to specify the location this library file.

The smart card is assumed to be in the card reader at one-time password generation time and a single card reader is also assumed.

If the user's smart card has not been authenticated the user is prompted to enter a PIN to unlock the card. This is required only once, because the PIN is normally cached.

5.2.27 GetCheckBoxState

Use with

Advanced Web Application Definition

SecureLogin Version	3.5 to 6.1 SP1
Type	Action
Usage	GetCheckBoxState <#Item Number> <Variable>
Arguments	<p><Item Number></p> <p>The ID of the check box.</p> <p><Variable></p> <p>The target variable for the status of the specified check box. The value returned is Checked or Unchecked. The variable can be a question mark (?) or a dollar sign (\$) variable.</p>
Description	Use the <code>GetCheckBoxState</code> command to return the current state of the specified check box.
Example	<pre>GetCheckBoxState #25 ?state1 GetCheckBoxState #26 ?state2 MessageBox ?state1 MessageBox ?state2</pre>

5.2.28 GetCommandLine

Use with	Startup, Windows
SecureLogin Version	3.0.4 to 6.1 SP1
Type	Action
Usage	GetCommandLine <Variable>
Arguments	<p><Variable></p> <p>This variable defines where to store the captured command line.</p>
Description	<p>Use the <code>GetCommandLine</code> command to capture the full command line of the program that is loaded, and save it to the specified variable.</p> <hr/> <p>NOTE: Use the <code>GetCommandLine</code> to detect and differentiate back-end systems and database for use with multiple logins in the SAP* application.</p> <p>This command is not supported under Windows 95 and Windows 98.</p> <hr/>
Example	<p>Windows Application Definition</p> <p>This example reads the command line of the application, and then tests the line to see if it is Notepad.exe. If it is, Notepad is closed. If it is not, the application definition ends.</p> <pre>GetCommandLine ?Text If ?Text Eq "C:\Winnt\notepad.exe" KillApp Notepad.exe EndIf</pre>

5.2.29 GetEnv

Use with	All
SecureLogin Version	3.5 to 6.1 SP1
Type	Action
Usage	GetEnv <envvar> <variable>
Arguments	<EnvVar> This is the environment variable name you want to retrieve. <variable> This variable defines where to store the retrieved environment variable data.
Description	Use the <code>GetEnv</code> command to read the value of an environment variable and saves it in the specified <variable>.
Example	Windows Application Definition <pre>GetEnv "SESSIONNAME" ?SessionName If ?SessionName eq "console" MessageBox "Running from Citrix Server Console" EndIf</pre>

5.2.30 GetHandle

Use With	Windows
SecureLogin Version	Introduced with Novell SecureLogin 6.1
Type	Action
Usage	GetHandle <variable>
Arguments	<Variable> This variable defines where to store the capture handle.
Description	Use <code>GetHandle</code> to capture the unique handle of the windows on which the Windows application definition script is activated.
Example 1	Windows Application Definition <pre>GetHandle ?winHandle MessageBox ?winHandle</pre>

Example 2**Windows Application Definition**

```
GetReg
"HKLM\Software\Microsoft\Windows\CurrentVersion\App
Paths\SLProto.exe\Path" ?SLLocation
If ?SLLocation eq "<NOTSET>"
EndScript
EndIf
GetHandle ?PuttyHWND
Strcat ?TLaunch ?SLLocation "tlaunch.exe"
Strcat ?TLaunchHWND "/hwnd" ?PuttyHWND
Run ?TLaunch "/auto" "/ePutty" "/l" "/pPutty - Detection
and Login" "/t" "/q" "/s" ?TLaunchHWND
```

5.2.31 GetIni

Use With	Windows, Web, Terminal, Java
SecureLogin Version	3.5 to 6.1 SP1
Type	Action
Usage	GetIni <ini file> <section> <key> <variable>
Arguments	<Ini File> This is the filename from which you want to read the section or key. <Section> Name of the section that contains the key name. <Key> Name of the key to read. <variable> This variable defines where to store the retrieved environment variable data.
Description	Use the GetIni command to read data from INI file.
Example	Windows Application Definition GetIni "c:\program files\lotus\notes\notes.ini" "Notes"? "KeyFileName ?NotesDefaultIDFileSetPlat ?NotesDefaultIDFile

5.2.32 GetMD5

Use With	Windows
SecureLogin Version	6.0
Type	Action
Usage	GetMD5 <value>

Arguments	<p><value></p> <p>Returns the MD5 hash value.</p>
Description	<p>Use the <code>GetMD5</code> command to generate an MD5 hash value of the current process the script is running for. <code>GetMD5</code> works only with Win32 scripts.</p> <p>Message-Digest algorithm 5 (MD5) is employed in SecureLogin and can be used to check the integrity of files against a known hash value.</p> <p>The MD5 hash is widely used in software to provide assurance that a particular file has not been altered. The administrator can compare a published MD5 sum with the checksum of another file to recognize corrupt or incomplete files, particularly for large executable files.</p>
Example	<p>In a Windows application definition the MD5 hash value is stored as a variable that is then passed in as the argument to the command, which could be a <code>?tmp</code> or <code>\$hash_value</code> type variable.</p> <pre>GetMD5 ?tmp</pre> <p>or</p> <pre>GetMD5 \$hash_value</pre> <p>The MD5 hash value is normally obtained from the Windows Finder tool on a window from the application, then the MD5 hash is copied from the Window Finder. This MD5 value is then be put in a script and the <code>GetMD5</code> command is used to compare the two MD5 hash values. If the MD5 hash values do not match, then the executable file might have been changed.</p>

5.2.33 GetReg

Use With	All
SecureLogin Version	3.5 to 6.1 SP1
Type	Action
Usage	<code>GetReg <regentry> <variable></code>
Arguments	<p><regentry></p> <p>This is the registry entry to read.</p> <p><variable></p> <p>This variable defines where to store the retrieved environment variable data.</p>

Description	<p>Use the <code>GetReg</code> command to read data from the registry and save it in the specified <code><variable></code>.</p> <p>The following is format for the registry entry input:</p> <pre>HIVE\KEY\Value</pre> <p>ValueValid hives are:</p> <pre>"HKCR" HKEY_CLASSES_ROOT"HKCC"HKEY_CURRENT_CONFIG"HKCU "HKEY_CURRENT_USER"HKLM"HKEY_LOCAL_MACHINE"HKU"HKEY_USERS</pre>
Example	<p>Windows Application Definition</p> <pre>GetReg "HKLM\Software\ABCCorp\ProductID"?ProductID If ?ProductID noteq "xxxxxxxxxx" #Not corporate desktop EndScript EndIf</pre>
Description	<p>To get the default value of a Registry key, use two backward slashes (<code>\\</code>) on the command.</p> <p>This returns the default value.</p>
Example	<pre>GetReg HKLM\XYZ\\</pre> <p>This returns the default value set at XYZ.</p>

5.2.34 GetSessionName

Use With	Terminal Emulator
SecureLogin Version	3.5 to 6.1 SP1
Type	Action
Usage	<code>GetSessionName <?variable></code>
Arguments	<p><code><Variable></code></p> <p>The target variable that the session name is copied into.</p>
Description	Used to find the current HLLAPI session name that is used to connect and return it to the specified variable.
Example	<p>Windows Application Definition</p> <pre>GetSessionName ?Session_name</pre>

5.2.35 GetText

Use With	Web, Terminal Launcher
SecureLogin Version	3.0 to 6.1 SP1
Type	Action

Usage	GetText <Variable>
Arguments	<Variable> This variable defines where to store the captured text.
Description	Use the <code>GetText</code> command to get all of the text from the screen and save it to the specified variable. It is used in a large Web application definition that might contain several If -Text statements. Under Netscape, each If -Text statement scans the screen to find the specified text, and each scan of the screen results in the screen flashing. However, by using <code>GetText</code> , (for example <code>If ?Text -in ?FromGetText</code>) the application definition can contain multiple If -Text commands with only one scan of the screen.
Example	Web Application Definition This example copies the text content of the Web page to the ?Text variable. SecureLogin tests for the presence of the words "Log on". If Log on exists, SecureLogin enters the credentials and submits them automatically. <pre>GetText ?Text If "Log on" -In ?Text Type \$Username Type \$Password Password EndIf</pre>

5.2.36 GetURL

Use With	Web
SecureLogin Version	3.0 to 6.1 SP1
Type	Action
Usage	GetURL <Variable>
Arguments	<Variable> This variable defines where to store the captured URL.
Description	Use the <code>GetURL</code> command to capture the URL of the site that is loaded and save it to the specified variable.
Example	Web Application Definition This example copies the URL of the Web site to the ?URL variable and tests the URL to see if it matches text being searched for. If it does, SecureLogin pops up a message box and redirects the user to the Intranet. <pre>GetURL ?URL If "Log off" -In ?URL MessageBox "You have chosen to log off the applications. You will now be redirected to the Intranet home page." GoToURL "http://Intranet" EndIf</pre>

5.2.37 GoToURL

Use with	Web
SecureLogin Version	3.5 to 6.1 SP1
Type	Action
Usage	GoToURL <URL> [<-frame>]
Arguments	<URL> The URL to which the browser navigates. <-frame> Opens the URL in the frame that started the application definition.
Description	Use the <code>GoToURL</code> command to make the browser navigate to the specified <URL>. By default, the command opens the new Web page in the main window, rather than the frame that started the application definition. When using the <code>-frame</code> option on a framed Web page, the URL redirect occurs only in the current frame rather than the parent window. You must specify <code>http://</code> before the URL.
Example	Web Application Definition This example detects an incorrect password message, displays a message box informing the user, and then browses the Web site. <pre>If -Text "Incorrect Password" MessageBox "You have entered an incorrect password" GoToURL "http://www.novell.com" EndIf</pre>

5.2.38 Highlight

Use with	Startup, Terminal Launcher, Web, or Windows
Novell SecureLogin version	3.5 or later
Type	Action
Description	Use the <code>Highlight</code> command to set the focus of the Web page on a field. The command is useful for pages that do not have any control selected after loading or for any fields that change the behavior after gaining focus. It functions similar to the <code>SetFocus</code> command in Windows scripts.

Example Web application definition

```
If -Text "Logon"
Highlight #1
Type $Username #1
Highlight #2
Type $Password #2
Type "\N"
EndIf
```

5.2.39 If/Else/EndIf

Use with Startup, Terminal Launcher, Web, or Windows

Novell SecureLogin version 3.5 or later

Type Flow control

Usage 1 If **<Value1>** <Gt|Lt> **<Value2>**
#Do This
[Else]
#Do This
EndIf

Usage 2 If **<Value1>** <Eq|NotEq > **<Value2>** [-I|-S]
#Do This
[Else]
#Do This
EndIf

Usage 3 If **<Value1>** <-In|-NotIn> **<Value2>** [-I|-S]
#Do This
[Else]
#Do This
EndIf

Usage 4 If -Text [-Frame] **<Text>**
#Do This
[Else]
#Do This
EndIf

Usage 5 If -Exists|-NotExists **<Variable>**
#Do This
[Else]
#Do This
EndIf

Arguments	<p><Value1></p> <p>The left side of the expression for evaluation.</p> <p><Value2></p> <p>The right side of the expression for evaluation.</p> <p><Text></p> <p>The text for which you are searching.</p>
Description	<p>Use the If command to establish a block to execute if the expression supplied is true. The Else command works inside an If block. The Else command is executed if the operator in the If block is false. Use the EndIf command to terminate the If block.</p> <p>Text comparison operators supported The text comparison operators supported by the If command are:</p> <ul style="list-style-type: none"> ◆ Eq: True if the left side is equal to the right side. ◆ NotEq: True if the left side is not equal to the right side. ◆ -In: True if the left side is a substring of the right side. ◆ -NotIn: True if the left side is not a substring of the right side. ◆ -SiteDeparted: Checks if the current document is still active or not. <p>When using these text comparison operators, you may optionally specify whether the comparison is to take into account the case of the strings being compared. If -I is specified, the comparison is case insensitive. If -S is specified, then the comparison is case sensitive. By default the Eq and NotEq operators are not case sensitive, while the -In and -NotIn operators are case sensitive.</p> <p>An operator is also supplied to directly query the application for a particular string:- Text: Evaluates to true if the specified text is found in the application windows of the application. For Internet Explorer application definitions, you can supply an optional -Frame argument, which restricts the command to look for the specified text in the current frame.</p> <p>Numerical comparison operators supported Two numerical comparison operators are supported by the If command, Gt and Lt. The command evaluates to true if the left side is greater than or less than (respectively) the right side. This is a numerical comparison, so the left and right sides must be numbers.</p> <p>An operator is supplied to check for the existence of a stored variable:</p> <ul style="list-style-type: none"> ◆ -Exists: True if the specified variable exists. ◆ -NotExist: True if the specified variable does not exist. <p>Syntax examples</p> <pre>If \$Number NotEq "1" MessageBox "NotEq 1" Else MessageBox "Eq 1" EndScript EndIf If ?Value1 Gt ?Value2 If -Text "Log on" If -Exists \$RunBefore If "Log on" -In ?Text</pre>

Example 1**Web application definition**

This example tests for an incorrect password. If it is found, an incorrect password message box is displayed. If the error message is not found, Novell SecureLogin logs in as normal.

```
If -Text "Incorrect Password"
DisplayVariables "You have an incorrect password. Please
verify it and retry log on."
EndScript
Else
Type $Username
Type $Password Password
EndIf
```

Example 2**Windows application definition**

Each time the application definition is run, a variable is incremented. This example counts the number of times the dialog box is displayed. If it is displayed more than three times, the application is closed. If the log on is successful, the count is reset.

```
# Logon Dialog Box
Dialog
  Class #32770
  Title "Log on"
EndDialog

ReadText #1001 ?Username

If -Exists $Username
Else
  Set $Username ?Username
EndIf
Increment ?RunCount
If ?RunCount Gt "3"
MessageBox "Log on has been attempted too many times. The
application will be closed."
KillApp "app.exe"

Else

  Type $Username #1001

  Type $Password #1002

  Click #1

EndIf

# Logon Successful Dialog Box
Dialog
  Ctrl #1
  Title "Log on successful"
EndDialog

Set ?RunCount "0"
```

Example 3**Web application definition**

This example copies the text content of the web page to ?WebText. The variable is then tested to see if 'Log on' is present. If it is, Novell SecureLogin performs the logon process. If it is not present, the application definition is terminated.

```
GetText ?WebText
If "Log on" -In ?WebText
    Type $Username
    Type $Password Password
Else
    EndScript
EndIf
```

Example 4**Startup**

This example tests, upon Novell SecureLogin loading, to see if Novell SecureLogin has been run by the user. If it has not, Novell SecureLogin sets the variable so that the message is only displayed once, and then displays a welcome message along with the option for further details on Novell SecureLogin.

```
If -NotExist $LoadedBefore
    EndScript
Else
    MessageBox -YesNo ?Result "Welcome to SecureLogin Single
    Sign-On, a new password management tool that will save you
    the hassle of remembering your passwords. Would you like more
    details on how to use SecureLogin and what it can do for
    you?"
    Set $LoadedBefore "Yes"
    If ?Result Eq "Yes"
        GoToURL "http://www.Novell.com/securelogin.htm"
    EndIf
EndIf
```

5.2.40 Include

Use With	All
SecureLogin Version	3.0 to 6.1 SP1
Type	Flow Control
Usage	Include <Platform-Name>
Arguments	<Platform-Name> The name of the application definition to include.
Description	Use the <code>Include</code> command to share commonly used application definition commands by multiple applications. The application definition identified by <Platform-Name> is included at execution time into the calling application definition. The application definition included with the <code>Include</code> command must comprise commands supported by the calling application.

Example	<p>Windows Application Definition</p> <p>This example detects the login dialog box, the <code>notepad.exe</code> application definition is executed, and then the user's credentials are entered.</p> <pre># Log on Dialog Box Dialog Class #32770 Title "Log on" EndDialog Include "Notepad.exe" Type \$Username #1001 Type \$Password #1002 Click #1</pre>
---------	---

5.2.41 Increment

Use With	All
SecureLogin Version	3.5 to 6.1 SP1
Type	Variable Manipulator
Usage	Increment <Variable>
Arguments	<p><Variable></p> <p>The name of the variable to increase in value.</p>
Description	<p>Use the <code>Increment</code> command to add to a specified variable. For example, you can use <code>Increment</code> to count the number of passes a particular application definition has made.</p> <p>After the number of instances is equal to the specified number, you can instruct the application definition to run another task or end the application definition. This is useful when configuring an application whose login panel is similar to other windows within the application, or to easily control the number of attempts a user can have to access an application.</p> <p>Also see Section 5.2.17, "Decrement," on page 79</p>
Syntax examples	Increment ?RunCount

Example**Windows Application Definition**

Each time the application definition is run, a variable is incremented. This example counts the number of times the dialog box is displayed. If the dialog box is displayed more than three times, the application is closed. If the log in is successful, the count is reset.

```
#Log on Dialog Box
Dialog
  Class #32770
  Title "Log on"
EndDialog
Increment ?RunCount
If ?RunCount Gt "3"
  MessageBox "Log on has been attempted too many
times. The
application will be closed."
  KillApp "app.exe"
Else
  Type $Username #1001
  Type $Password #1002
Click #1
EndIf

# Log on Successful Message
Dialog
  Ctrl #1
  Title "Log on Successful"
EndDialog
Set ?RunCount "0"
```

5.2.42 KillApp

Use With	All
SecureLogin Version	3.5 to 6.1 SP1
Type	Action
Usage	KillApp <Process-Name>
Arguments	<Process-Name> The name of the process to terminate.
Description	Use to terminate an application.

Example**Windows Application Definition**

Each time the application definition is run, a variable is incremented. This example counts the number of times the dialog box is displayed. If the dialog box is displayed more than three times, the application is closed. If the login is successful, the count is reset.

```
#Log on Dialog Box
Dialog
    Title "Log on"
    Class #32770
EndDialog

Increment ?RunCount
If ?RunCount Gt "3"
    MessageBox "Log on has been attempted too many times.
The application will be closed."
    KillApp "app.exe"
Else
    Type $Username #1001
    Type $Password #1002
    Click #1
EndIf

# Log on Successful Message
Dialog
    Title "Log on Successful"
    Ctrl #1
EndDialog

Set ?RunCount "0"
```

5.2.43 Local

Use with	All
SecureLogin Version	3.5 to 6.1 SP1
Type	Variable Manipulator
Usage	Local <?Variable>
Arguments	<?Variable>
	The runtime variable to declare as local.
Description	<p>Use the <code>Local</code> command to declare that a runtime variable only exists for the lifetime of the application definition. Local runtime variables are used in the same way as normal runtime variables and are still written as <code>?Variable</code>.</p> <p>Declare local runtime variables as local by using the <code>Local</code> command, followed by the variable name. When runtime variables are declared local, you cannot set them back again. You can declare a runtime variable local at any time in an application definition.</p>

Description	<p>Using local runtime variables increases the performance of SecureLogin, although only slightly. Local runtime variables are used to run application definitions multiple times and not store the runtime variables between each run of the application definition.</p> <p>Local runtime variables are also used to prevent runtime variables from overwriting each other, which could happen if two instances of an application definition are running at the same time. For example, use the <code>Local</code> command if two instances of Terminal Launcher are running, each instance running the same application definition, but attached to different emulator sessions.</p>
Example	<p>Windows Application Definition</p> <p>This example declares a variable as local, and then uses it to count the number of times a dialog box is displayed. If the dialog box is displayed too many times, SecureLogin alerts the user, then closes the application.</p> <pre> # Invalid Log on Message Dialog Class #32770 Title "Log on Failure" EndDialog Local ?RunCount Increment ?RunCount If ?RunCount Gt "5" MessageBox "Closing Application" KillApp "PasswordText.exe" EndIf Type \$Username Type \$Password </pre>

5.2.44 MatchDomain

Use With	Advanced application definitions created by using the Web Wizard.
SecureLogin Version	3.6.1.0 to 6.1 SP1
Type	Action
Usage	MatchDomain "Domain"
Arguments	<p>Domain</p> <p>The Domain name or address to be matched.</p>
Description	<p>Use MatchDomain inside a site block to filter a Site based on its domain. If the domain doesn't match, the site block fails to match.</p> <p>The domain matched is a normally a low-level domain name such as <code>www.yahoo.com</code> and not <code>http://www.yahoo.com/mymail/login</code>.</p>

Example

This example the Web site www.google.com is being matched by the Application Definition.

```
# === Login Application Definition #2 ==
# === Google Initial Login ===
#=====
Site Login -userid "Google Log On" -initial
MatchDomain "www.google.com"
MatchField #1:1 -name "Email" -type "text"
MatchField #1:2 -name "Passwd" -type "password"
MatchField #1:3 -name "Cookie" -type "check"
EndSite
SetPrompt "Enter your user credentials"
TextInput #1:1 -value "$Username"
TextInput #1:2 -value "$Password"
FocusInput#1:2 -focus "true"
BooleanInput #1:3 -check "false"
PressInput
Endscript
```

5.2.45 MatchField

Use With	Advanced application definitions created by using the Web Wizard.
SecureLogin Version	3.6.1.0 to 6.1 SP1
Type	Action
Usage	MatchField #FormID:FieldID [-optional] [-name "name"] [-type "type"] [-value "value"] [-defaultValue "defaultValue"]

Arguments	<p>FieldID</p> <p>The ID to be given to the matched field. The ID must be a static unsigned integer.</p> <p>-optional</p> <p>Specifies that matching this field is not required to successfully match the parent form.</p> <p>-name "name"</p> <p>Match against the field name.</p> <p>-type "type"</p> <p>Match against the field type. Type can be one of the following:</p> <ul style="list-style-type: none"> ◆ Button ◆ Checkbox ◆ File ◆ Image ◆ Hidden ◆ Password ◆ Radio ◆ Reset ◆ Submit ◆ Text ◆ Select-multiple ◆ Select-one <p>-value "value"</p> <p>Match against the field value.</p> <p>-defaultValue "defaultValue"</p> <p>Match against the field's default value.</p>
Description	<p>Use MatchField to filter a form based on the presence of a particular field. If the field fails to match and it is not specified as optional, then the parent form fails to match.</p>

Example

This example would locate the Web site fields *Email*, *Password*, and *Cookie* within the Web site *www.google.com*, and matches *.com* in the application definition.

```
# === Login Application Definition #2 ==
# === Google Initial Login ===
#=====
Site Login -userid "Google Log On" -initial
MatchForm #1 -name "log on"
MatchDomain "www.google.com"
MatchField #1:1 -name "Email" -type "text"
MatchField #1:2 -name "Passwd" -type "password"
MatchField #1:3 -name "Cookie" -type "check"
MatchField #1:4 -name "SAVEOPTION" -type "checkbox" -
value "YES"
MatchField #1:5 -name "Submit2" -type "submit"

EndSite
SetPrompt "Enter your user credentials"
TextInput #1:1 -value "$Username"
TextInput #1:2 -value "$Password"
FocusInput#1:2 -focus "true"
BooleanInput #1:3 -check "false"
BooleanInput #1:4 -check "false"

PressInput
Endscript
```

5.2.46 MatchForm

Use With	Advanced application definitions created by using the Web Wizard.
SecureLogin Version	3.6.1.0 to 6.1 SP1
Type	Action
Usage	MatchForm #FormID [-optional] [-name "name"] [-action "action"] [-method "method"] [-target "target"]

Arguments	<p>FormID</p> <p>The ID to be given to a matching form. The ID must be a static unsigned integer.</p> <p>-optional</p> <p>Specifies that matching this form is not required to successfully match the site.</p> <p>-name "name"</p> <p>Specifies the form name to match against. The form name is an optional value given to a form by the creator of the Web site.</p> <p>-action "action"</p> <p>Specifies the form action to match against. The URL to which the form content is sent for processing (MDSN definition).</p> <p>-method "method"</p> <p>Specifies the form method to match against. The method or how to send the form data to the server (MDSN definition.)</p> <p>-target "target"</p> <p>Specifies the form target to match against. The window or frame at which to the form targets its contents.(MDSN definition).</p>
Description	<p>Use <code>MatchForm</code> to filter a site based on the presence of a particular form. If the form fails to match and it is not specified as optional, then the site fails to match.</p>
Example	<p>This example the form name <code>log on</code> within the Web site <code>www.google.com</code> and <code>.com</code> is being matched by the application definition.</p> <pre data-bbox="552 1155 1218 1680"> # === Login Application Definition #2 == # === Google Initial Login === #===== Site Login -userid "Google Log On" -initial MatchForm #1 -name "log on" MatchDomain "www.google.com" MatchField #1:1 -name "Email" -type "text" MatchField #1:2 -name "Passwd" -type "password" MatchField #1:3 -name "Cookie" -type "check" EndSite SetPrompt "Enter your user credentials" TextInput #1:1 -value "\$Username" TextInput #1:2 -value "\$Password" FocusInput#1:2 -focus "true" BooleanInput #1:3 -check "false" PressInput Endscript </pre> <p>The form name can be a "null"</p> <pre data-bbox="552 1722 893 1764"> MatchForm #1 -name "" </pre>

5.2.47 MatchOption

Use With	Advanced Web application definitions created by using the Web Wizard.
SecureLogin Version	3.6.1.0 to 6.1 SP1
Type	Action
Usage	MatchOption #FormID:FieldID:OptionID [-optional] [-text "text"] [-value "value"]
Arguments	<p>OptionID</p> <p>The ID to be given to the specific option within the given field. The ID is a static, unsigned integer.</p> <p>-optional</p> <p>Specifies that matching this option is not required to successfully match the parent field.</p> <p>-text "text"</p> <p>Specifies the text string for this particular option.</p> <hr/> <p>NOTE: The text is what is displayed to the user.</p> <hr/> <p>-value "value"</p> <p>Specifies the value for this particular option.</p> <hr/> <p>NOTE: The value is what is passed to the server when a form is submitted.</p> <hr/>
Description	<p>Use the <code>MatchOption</code> command to filter a field based on the presence of a particular option.</p> <p>An option is an item within a specific combo box or list box. If the option is not found, and it is not specified as optional, then the parent field also fails to match.</p>

Example

This example uses the form name "log on" within the secure Web site www.lotto.com and .com is being matched by the application definition.

```
# === Login Application Definition #4 ==
# === Lotto User Initial Login ====
#=====
Site Login -userid "Member Log In" -initial
MatchForm #1 -name "log in"
MatchDomain "https://site10.Lotto.com"
MatchField #1:1 -name "Member ID" -type "text"
MatchField #1:2 -name "Passwd" -type "password"
MatchOption #1:3 -name "Secure" -type "text"
EndSite
SetPrompt "Enter your user credentials"
TextInput #1:1 -value "$Username"
TextInput #1:2 -value "$Password"
FocusInput #1:2 -focus "true"
BooleanInput #1:3 -check "true"
PressInput
Endscript
```

5.2.48 MatchReferer

Use With	Advanced Web application definitions created by using the Web Wizard.
SecureLogin Version	3.6.1.0 to 6.1 SP1
Type	Action
Usage	MatchReferer "Referer"
Arguments	MatchReferer Used inside a site block, MatchReferer is used to filter a Site based on a referer. If the site referer does not match, the site block fails to match. "Referer" The site referer which is to be matched. If PageA.htm includes a link to PageB.htm, then the referer is "PageA.htm".
Description	Use MatchReferer inside a Site/EndSite block to match or filter a Site based on a referrer.

Example In this example, the referring html page "www.lotteries/index.html" is being matched by the application definition.

```
# === Login Application Definition #5 ==
# === Lotto User Initial Login ====
#=====
Site Login -userid "Member Log In" -initial
MatchForm #1 -name "log in"
MatchReferer "www.Lotteries.com/index.html"
MatchDomain "https://site10.Lotto.com"
MatchField #1:1 -name "Member ID" -type "text"
MatchField #1:2 -name "Passwd" -type "password"
MatchOption #1:3 -name "Secure" -type "text"
EndSite
SetPrompt "Enter your user credentials"
TextInput #1:1 -value "$Username"
TextInput #1:2 -value "$Password"
FocusInput #1:2 -focus "true"
BooleanInput #1:3 -check "true"
PressInput
Endscript
```

5.2.49 MatchTitle

Use With	Advanced Web application definitions created by using the Web Wizard.
SecureLogin Version	3.6.1.0 to 6.1 SP1
Type	Action
Usage	MatchTitle "Title"
Arguments	MatchTitle Used inside a site block, MatchTitle is used to filter a Site based on its title. If the site title does not match, the site block fails to match. "Title" The site Title that is to be matched.
Description	Use MatchTitle inside a site block to match or filter a site based on an HTML page title.

Example This example the html page with the title "" within the Web site www.nytimes.com .com is being matched by the application definition.

```
# =====  
# ===== Login Script #1 - The New York Times > Log  
In  
# =====  
# === Initial Login ===  
Site Login -userid "nytimes.com #1" -initial  
  MatchURL "http://www.nytimes.com/auth/login"  
  MatchDomain "www.nytimes.com"  
  MatchTitle "The New York Times > Log In"  
  MatchForm #1 -name "login"  
  MatchField #1:1 -name "USERID" -type "text"  
  MatchField #1:2 -name "PASSWORD" -type "password"  
  MatchField #1:3 -name "SAVEOPTION" -type "checkbox"  
-value "YES"  
  MatchField #1:4 -name "Submit2" -type "submit"  
EndSite
```

5.2.50 MatchURL

Use With	Advanced Web application definitions created by using the Web Wizard
SecureLogin Version	3.6.1.0 to 6.1 SP1
Type	Action
Usage	MatchURL "URL"
Arguments	MatchURL Used inside a site block, MatchURL is used to filter a Site based on its URL. If the URL doesn't match, the site block fails to match. "URL" The Site URL that is to be matched. This need not be the URL listed in the navigation field of the Web browser because the given page might not have been loaded from there.
Description	Use Match URL inside a site block to match or filter an HTML page within a Site based on its URL. The URL can be a complex Web address or a secure Web site such as: https://mymail.com/home/login.php?Home=eb9127d7df248d0e63d92

Example In this example, the URL "https://www.nytimes.com/auth/login" is matched.

```
# === Initial Login ===
Site Login -userid "nytimes.com #1" -initial
  MatchURL "https://www.nytimes.com/auth/login"
  MatchDomain "www.nytimes.com"
  MatchTitle "The New York Times > Log In"
  MatchForm #1 -name "login"
  MatchField #1:1 -name "USERID" -type "text"
  MatchField #1:2 -name "PASSWORD" -type "password"
  MatchField #1:3 -name "SAVEOPTION" -type "checkbox"
  -value "YES"
  MatchField #1:4 -name "Submit2" -type "submit"
EndSite
```

5.2.51 MessageBox

Use With	Startup, Terminal Launcher, Web, or Windows
SecureLogin Version	3.5 to 6.1 SP1
Type	Action
Usage	MessageBox <Data> [-Background] [-DefaultNo] [-YesNo <?Variable>] [-YesNoCancel <?Variable>]
Arguments	<p><-YesNo></p> <p>The -YesNo flag allows the user to select <i>Yes</i> or <i>No</i> within the message box, rather than being limited to an OK button only.</p> <p><-YesNoCancel></p> <p>The -YesNoCancel flag allows the user to select <i>Yes</i>, <i>No</i>, or <i>Cancel</i> when a message box is displayed.</p> <p><?Variable></p> <p>This runtime variable is required with the -YesNo / -YesNoCancel flag to store the result of the user action.</p> <p><-Background></p> <p>When specified, this parameter 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. In Web applications, you must respond to the MessageBox before you can continue with any other work.</p> <p><-DefaultNo></p> <p>This optional parameter is used only with the -YesNo and -YesNoCancel flags. When the -DefaultNo parameter is set, the <i>No</i> button has the default focus rather than the <i>Yes</i> button.</p> <p><Data></p> <p>The text displayed to the user. <Data> can be several strings, variables, or a combination of both.</p>

Description	<p>Use the <code>MessageBox</code> command to display a dialog box that contains the text specified in the <code><Data></code> variable. The application definition is suspended until the user reacts to this message. The <code>MessageBox</code> can take any number of text arguments, including variables, (for example <code>MessageBox "The user " \$Username " has just been logged in to the system"</code>).</p> <p>You can set the <code>-YesNo</code> flag when calling a <code>MessageBox</code>. If the <code>-YesNo</code> flag is set, the <code>MessageBox</code> prompts the user with a box that has a <code>Yes</code> and a <code>No</code> button, rather than an <code>OK</code> button.</p> <p>Use a runtime <code><?Variable></code> to capture the <code>MessageBox</code> result immediately after the flag. The variable value is set to <code>Yes</code>, <code>No</code>, or <code>Cancel</code>.</p>
Syntax examples	<pre> MessageBox "Application Definition completed successfully" MessageBox "Do you wish to continue?" -YesNo ?Result MessageBox "Do you wish to continue?" -YesNoCancel ?Result -Background - DefaultNo </pre>
Example 1	<p>Windows Application Definition</p> <p>This example detects the Change Password dialog box. A message box is displayed prompting the user whether or not they want to change their password, and to inform them it was successful.</p> <pre> # Change Password Dialog Box Dialog Class #32770 Title "Change Password" EndDialog MessageBox -YesNo ?Result "Your password has expired, would you like to change it now?" If ?Result Eq "Yes" Type \$Username #1015 Type \$Password #1004 ChangePassword \$Password Random Type \$Password #1005 Type \$Password #1006 Click #1 MessageBox "Password changed successfully" Else Click #2 MessageBox "You elected not to change your password." EndIf </pre>

Example 2**Terminal Launcher Test Application Definition**

Use message boxes when troubleshooting application definitions. This example displays a message box before each step in the application definition to allow the writer to see where the application definition execution is failing.

The `WaitForText` cuts off the first character because it finds both `Password` and `password`, and responds to all password entry points.

```
MessageBox "Beginning wait for Log on prompt"
WaitForText "login:"
MessageBox "Log on detected, now entering Username"
Type $Username
MessageBox "Username entered, now simulating Enter"
Type @E
MessageBox "Enter has been simulated. Now waiting for?
Password"WaitForText "password:"
MessageBox "Password detected, now entering Password"
Type $Password
MessageBox "Password entered, now simulating Enter"
Type @E
MessageBox "Sequence completed, the user should now be
logged on"
```

5.2.52 Multiply

Use With	All
SecureLogin Version	3.0 to 6.1 SP1
Type	Variable Manipulator
Usage	Multiply <Variable1> <Variable2> [?Result]

NOTE: You must use integer arithmetic.

Arguments

<Variable1>

The `Multiply` command, which is the first argument, is the number multiplied by the second argument. Also this argument contains the result if the optional `[?Result]` argument is not passed in. If used without the `[?Result]` argument, `<Variable1>` must be a SecureLogin variable, either `?Variable1` or `$Variable1`. Otherwise `<Variable1>` can be any numeric value.

<Variable2>

The multiplier, which is the second argument, is the number by which the first number is multiplied. `<Variable2>` can be a SecureLogin variable or numeric value.

[?Result]

Optional. The product, or result of the equation.

Description

Use to multiply one number by another. You can hard code the numbers into the application definition, or you can use variables. The results can be output to another variable, or to one of the original numbers.

Syntax examples	<pre>Multiply "1" "2" ?Result Multiply ?LoginAttempts ?LoginFailures Multiply ?LoginAttempts ?LoginFailures ?Result Multiply ?LoginAttempts "3" Multiply ?LoginAttempts "3" ?Result</pre>
Example	<p>Windows Application Definition</p> <p>This example reads the values of Control IDs 103 and 104 into variables. From there they are multiplied, and typed into Control ID 1.</p> <pre>ReadText #103 ?Number1 ReadText #104 ?Number2 Multiply ?Number1 ?Number2 ?Result Type ?Result #1</pre>

5.2.53 OnException/ClearException

Use With	All
SecureLogin Version	3.0.4 to 6.1 SP1
Type	Flow Control
Usage	<pre>OnException <Exception Name> Call <SubRoutine> ClearException <Exception Name></pre>

Arguments	<p><Exception Name></p> <p>The name of the exception on which you want to act. The following exceptions are supported:</p> <ul style="list-style-type: none"> ◆ <code>AAVerifyCancelled</code>. When a user cancels the reauthentication process (support depends on the Advanced Authentication product being used). ◆ <code>AAVerifyFailed</code>. When the <code>AAVerify</code> reauthentication command fails. ◆ <code>ChangePasswordCancelled</code>. When a user cancels in the <i>Change Password</i> dialog box. ◆ <code>EnterVariablesCancelled</code>. When a user cancels the automatic variable prompt box or the display variables prompt box. ◆ <code>GenerateOTPCancelled</code>. When a user cancels the one-time password generation dialog. ◆ <code>GenerateOTPFailed</code>. When the <code>GenerateOTP</code> command fails. ◆ <code>PickListCancelled</code>. When a user cancels the pick list choice dialog box. ◆ <code>RunFailed</code>. When the program specified by the <code>Run</code> command fails to launch. <p><SubRoutine></p> <p>The name of the subroutine you want to run when the exception condition is true.</p>
Description	<p>Use the <code>OnException</code> command to detect when certain conditions are met. Currently, this is when <i>Cancel</i> is clicked in either of two dialog boxes. When the condition is met, a subroutine is run. Use the <code>ClearException</code> command to reset the exceptions value.</p>
Syntax examples	<pre>OnException ChangePasswordCancelled Call Display Error ClearException ChangePasswordCancelled</pre>

Example 1**Windows Application Definition**

In this example, the login failed because the user has invalid credentials stored. This provides the user with an opportunity to verify his or her username and password, but if the user clicks *Cancel*, the exception is executed and forces the user to enter the credentials.

```
# Log on Failed Dialog Box
Dialog
  Class #32770
  Title "Log on Failed"
EndDialog
OnException EnterVariablesCancelled Call Variables
Cancelled
DisplayVariables "Please verify your Username and
Password and try again. Helpdesk x5555."
ClearException EnterVariablesCancelled

Type $Username #1001
Type $Password #1002
Click #1
Sub VariablesCancelled
  OnException EnterVariablesCancelled Call Variables
  Cancelled
  Display Variables "You cannot cancel this
verification dialog box. Please verify your Username
and Password when prompted and click OK to retry log
on."
  ClearException EnterVariablesCancelled
EndSub
```

Example 2**Windows Application Definition**

This example prompts the user to change his or her password. SecureLogin must handle password changes so the password is updated both in the application and in the user's 3DES encrypted store (in the directory against the user object).

```
# Change Password Dialog Box
Dialog
  Class #32770
  Title "Change Password"
EndDialog

Type $Username #1005
Type $Password #1006
OnException ChangePasswordCancelled Call ForceChangePwd
ChangePassword $Password "Please enter a new password
for the Human Resources? application. IT x5555"
Type $Password #1007
Type $Password #1008
ClearException ChangePasswordCancelled

Sub ForceChangePwd
  OnException ChangePasswordCancelled Call
ForceChangePwd
  ChangePassword $Password "You must enter a new
password and cannot Cancel.?
IT x5555"
  Type $Password #1007
  Type $Password #1008
  ClearException ChangePasswordCancelled
EndSub
```

Example 3**Windows Application Definition**

This example demonstrates the OnException usage of AAVerifyCancelled and AAVerifyFailed.

```
#
# Login - Simple
#
Dialog
  Title "Login - Simple"
  Class "#32770"
  Ctrl #1001
  Ctrl #1002
  Ctrl #1 "&Login"
  Ctrl #2 "Cancel"
  Ctrl #1027 "Username:"
  Ctrl #1028 "Password:"
  Ctrl #1009
EndDialog
  OnException AAVerifyCancelled Call
CancelSimpleLoginDialogCancelled
  OnException AAVerifyFailed Call
CancelSimpleLoginDialogFailed
  AAVerify -method "smartcard"
  Type $Username #1001
  Type $Password #1002
  Click #1
#
# Cancel the Simple Login Window - AAVerify cancelled
#
Sub CancelSimpleLoginDialogCancelled
  Click #2
  EndScript
EndSub
#
# Cancel the Simple Login Window - AAVerify failed
#
Sub CancelSimpleLoginDialogFailed
  Click #2
  MessageBox "Your re-authentication failed. Login
cancelled"
  EndScript
EndSub
```

Example 4**Windows Application Definition**

This example demonstrates the OnException usage of GenerateOTPCancelled and GenerateOTPFailed.

```
#
# Login - Simple
#
Dialog
  Title "Login - Simple"
  Class "#32770"
  Ctrl #1001
  Ctrl #1002
  Ctrl #1 "&Login"
  Ctrl #2 "Cancel"
  Ctrl #1027 "Username:"
  Ctrl #1028 "Password:"
  Ctrl #1009
EndDialog
  OnException GenerateOTPCancelled Call
CancelSimpleLoginDialogCancelled
  OnException GenerateOTPFailed Call
CancelSimpleLoginDialogFailed
  GenerateOTP -mode "AISC-SKI" ?OtpResult
  Type $Username #1001
  Type ?OtpResult #1002
  Click #1
#
# Cancel the Simple Login Window - GenerateOTP
cancelled
#
Sub CancelSimpleLoginDialogCancelled
  Click #2
  EndScript
EndSub
#
# Cancel the Simple Login Window - GenerateOTP failed
#
Sub CancelSimpleLoginDialogFailed
  Click #2
  MessageBox "Your generation of your password failed.
Login cancelled"
  EndScript
EndSub
```

5.2.54 Parent/EndParent

Use With	Windows
SecureLogin Version	3.5 to 6.1 SP1
Type	Dialog Specifier
Usage	ParentEnd
	Parent

Arguments	None
-----------	------

Description	Use the <code>Parent</code> command to begin a parent block in which the statements act upon a window's parent. The commands that follow the <code>Parent</code> command function identically to commands used in a dialog block; if they equate to false, then the application definition 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 block, or the application definition terminates if no further dialog blocks exist.

The `Parent` command is particularly useful in applications where the dialog box (for example login dialog box) is the child of an open window, typically in the background. If you are unable to single sign-on to an application after enabling it with the wizard, you typically need to specify parent blocks.

You can also use the `Parent` command to execute commands on a dialog box's parent. For example, it is possible to get an application definition to click a button on the parent window. An example of this use is shown in ["Example 2" on page 119](#).

EndParent Command Use the `EndParent` command to terminate a `Parent` block and set the subject of the application definition back to the original window. You can nest the `Parent` command, thereby allowing the parent block to act on the parent of the parent.

NOTE: If you use the wizard or try to enable an application and it does not seem work, try by using the `Parent` command. It is able to handle windows that are within windows, and so on.

Example 1

Windows Application Definition

This example specifies the dialog box that is used for login. In this case, the parent of the login box has a class of "Centura:MDIFrame".

```
#Log on Dialog BoxDialog
Class "Centura:Dialog"
  Ctrl #4098
  Ctrl #4100
  Title "Log on"
  Parent
    Class "Centura:MDIFrame"
  EndParent
EndDialog

Type $Username #4098
Type $Password #4100
Click #4101
```

Example 2**Windows Application Definition**

This example is used to click a button on the login window's parent.

```
# Log on Dialog Box
Dialog
  Class #32770
  Title "Log on"
EndDialog

Type $Username #1001
Type $Password #1002
Parent
  Click #1
EndParent
```

5.2.55 PickListAdd

Use With	All
SecureLogin Version	3.5 to 6.1 SP1
Type	Action
Usage	PickListAdd <Display-Text> [<Return-Value>]
Arguments	<Display-Text> The text displayed in the pick list for the specified option. <Return-Value> The value returned from the pick list. If a value is not specified, the return value is the display text.
Description	<p>Use the <code>PickListAdd</code> command to allow users with multiple accounts for a particular system to choose the account to which they login.</p> <p>You can also use the <code>PickListAdd</code> command to choose from multiple sessions on one mainframe account. In fact, you can use the <code>PickList</code> to build a list of databases, phone numbers, or any list from which your user can choose. You can then set variables or take action accordingly.</p> <p><code>PickListAdd</code> is always used with the <code>PickListDisplay</code> and is typically also used in conjunction with the <code>SetPlat</code> command.</p> <hr/> <p>NOTE: This command changed in usage from Novell SecureLogin 6.0 to 6.1. Setting variables after adding them to the list no longer results in the new value appearing in the list.</p> <p>For example,</p> <pre>PickListAdd ?Y Set ?Y "Text" PickListDisplay ...</pre> <p>displays the value <not set></p>

Example 1**Windows Application Definition**

In this example, the user must pick which of three accounts to use. They pick which account they want to use, and SecureLogin switches to that set of credentials by using the `SetPlat` command.

```
###Login Dialog Box
Dialog
Class #32770
Title "Log on"
EndDialog
PickListAdd "Account One" "One"
PickListAdd "Account Two" "Two"
PickListAdd "Account Three" "Three"
PickListDisplay ?Account "Please select the account you
wish to use"-NoEdit
SetPlat ?Account
Type $Username #1001
Type $Password #1002
Click #1
###End Login Dialog Box
```

Example 2**Any Application Definition**

In this example, the application should execute when Novell SecureLogin runs. It should display the numbers 1 - 10.

```
Set ?Count "1"
Repeat 10
PickListAdd ?Count
Increment ?Count
EndRepeat
PickListDisplay ?Count "Please select your option " -
NoEdit
```

Example 3**Windows Application Definition**

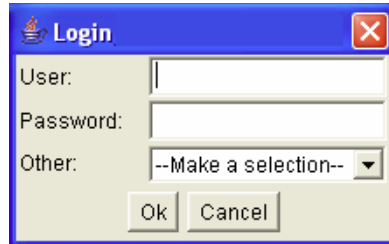
In this example, SecureLogin SSO reads the Control ID for #1001. (Use Windows Finder to find the Control ID.) SecureLogin should then display everything that is listed in that Control ID.

```
###Start Login
Dialog
Title "Login"
EndDialog
PickListAdd #1003
PickListDisplay ?Result "Select Database or Server"
SetPlat ?Result
Type $Username #1001
Type $Password #1002
Select ?result #1003
###End Login
```

Example 4**Java Application Definition**

In this example, when SecureLogin SSO runs it reads the Control ID for #1001. (Use Windows Finder to find the Control ID.) SecureLogin SSO should then Display everything that is listed in that Control ID.

In this example, when SecureLogin runs it must display



Create the Java Application Login_User_Pass_Other_List.exe with the following code:

```
PickListAdd #3
PickListDisplay ?Database "Select your Database" -
noedit
SetPlat ?Database
Type #1 $Username
Type #2 $Password
Select ?Database #3
###End Login##
```

Example 5**Windows Application Definition**

In this example, when SecureLogin runs it displays what you select.

If you Select #Fred, then the message box should display one thing, and if you select #1 then the message box should display other thing. Create the Windows Application ListComboTest.exe with the following code:

```
###Start ListComboTest
dialog
title "ListTest"
enddialog
PickListAdd #Fred something
PickListAdd #1 otherthing
PickListDisplay ?Result "my message"
Messagebox ?Result
###End ListComboTest##
```

5.2.56 PickListDisplay

Use With	Startup, Terminal Launcher, Web, and or, or Windows
SecureLogin Version	3.5 to 6.1 SP1
Type	Action
Usage	PickListDisplay <?Variable> <Display-Text> [-NoEdit]

Arguments	<p><?Variable></p> <p>The output variable for the selected option.</p> <p><Display-Text></p> <p>The description text for the pick list box.</p> <p>-NoEdit</p> <p>The -NoEdit flag disables the addition of extra variables by the user.</p>
Description	<p>Use the <code>PickListDisplay</code> command to display the pick list entries built by previous calls to <code>PickListAdd</code>. The <code>PickListDisplay</code> command returns the result in a <?Variable> sent to the command.</p> <p>If the desired entry is not among the displayed entries, the user can enter his or her own data into an edit field at the bottom of the pick list. Set the -NoEdit flag to turn this feature off.</p>
Syntax examples	<pre>PickListDisplay ?Choice "Please select the account you wish to use" PickListDisplay ?Choice "Please select the account you wish to use" -NoEdit</pre>
Example	<p>Windows Example</p> <p>In this example, the user has three accounts for this application, and wants to pick which one to use. After he or she picks which account they want to use, and SecureLogin uses the <code>SetPlat</code> command to switch to that set of credentials.</p> <pre># Log on Dialog Box Dialog Class #32770 Title "Log on" EndDialog PickListAdd "Account One" "One" PickListAdd "Account Two" "Two" PickListAdd "Account Three" "Three" PickListDisplay ?Account "Please select the account you wish to use" -NoEdit SetPlat ?AccountType \$Username #1001 Type \$Password #1002 Click #1</pre>

5.2.57 PositionCharacter

Use With	Password Policy application definitions
SecureLogin Version	3.5 to 6.1 SP1
Type	Action
Usage	POSITIONCHARACTER [NUMERAL] [UPPERCASE] [LOWERCASE] [PUNCTUATION] <Position>, [<Position>].

Arguments	<p>[NUMERAL]</p> <p>The character at <Position> must be a numeral.</p> <p>[UPPERCASE]</p> <p>The character at <Position> must be an uppercase character.</p> <p>[LOWERCASE]</p> <p>The character at <Position> must be a lowercase character.</p> <p>[PUNCTUATION]</p> <p>The character at <Position> must be a punctuation character.</p> <p><Position></p> <p>The character position in the password.</p>
Description	<p>Use this command in a password policy application definition to enforce that a certain character in the password is a numeral, uppercase, lowercase, or a punctuation character.</p> <p>You can specify multiple positions.</p>
Example	<p>The password is not valid unless the first, sixth, and seventh characters are uppercase.</p> <pre>POSITIONCHARACTER UPPERCASE 1,6,7</pre>

5.2.58 PressInput

Use With	Advanced Web application definitions created by using the Web Wizard.
SecureLogin Version	3.5 to 6.1 SP1
Type	Action
Usage	PressInput [#FormID:FieldID [-press "press"]]
Arguments	<p>PressInput</p> <p>Simulates a keyboard enter event, optionally focusing a given field beforehand.</p> <p>-press "press"</p>
Description	Simulates pressing the keyboard enter key.

Example

This example the `PressInput` command within the application definition is the equivalent of clicking the *Sign On* button on the `www.google.com` Web site.

```
# === Login Application Definition #2 ==
# === Google Initial Login ===
#=====  
# === Login Application Definition #2 ==  
# === Google Initial Login ===  
#=====  
Site Login -userid "Google Log On" -initial  
MatchForm #1 -name "log on"  
MatchDomain "www.google.com"  
MatchField #1:1 -name "Email" -type "text"  
MatchField #1:2 -name "Passwd" -type "password"  
MatchField #1:3 -name "Cookie" -type "check"  
EndSite  
SetPrompt "Enter your user credentials"  
TextInput #1:1 -value "$Username"  
TextInput #1:2 -value "$Password"  
FocusInput#1:2 -focus "true"  
BooleanInput #1:3 -check "false"  
PressInput  
Endscript
```

5.2.59 ReadText

Use With	Terminal Launcher, Windows. This command applies specifically to HLLAPI, WinHLLAPI and HLLAPI 16 terminal emulators.
SecureLogin Version	3.5 to 6.1 SP1
Type	Action
Windows Usage	<code>ReadText <#Ctrl-ID> <?Variable></code>
Terminal Launcher Usage	<code>ReadText <?Variable> <Character-Number> <Row-Number> <Column-Number></code>

Arguments	<p><#Ctrl-ID></p> <p>The control ID number of the text to read.</p> <p><?Variable></p> <p>The variable that receives the text that is read.</p> <p><Character-Number></p> <p>The number of characters to read.</p> <p><Row-Number></p> <p>The horizontal position number of the first character to read (for example, row).</p> <p><Column-Number></p> <p>The vertical position number of the first character to read (for example, column).</p>
Description	<p>Use the <code>ReadText</code> command to run in both Windows and Terminal Launcher application definitions. Although the usage and arguments for the use of <code>ReadText</code> with Windows and Terminal Launcher are different, the results of each command are the same.</p> <p>Windows Application Definition: In a Windows application definition, the <code>ReadText</code> 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.</p> <p>Terminal Launcher Application Definition: In a Terminal Launcher application definition, the <code>ReadText</code> command reads a specified number of characters, starting at the <Row-Number>, and sends those characters to the specified <Variable>. The <code>ReadText</code> command does not work with Generic or Advanced Generic emulators, it only works with HLLAPI and some DDE emulators. For Generic or Advanced Generic emulators, use the <code>If -Text</code> or <code>Gettext</code> commands.</p> <p>For more information, see Section 5.2.39, "If/Else/EndIf," on page 97 and Section 5.2.35, "GetText," on page 94.</p>
Example 1	<p>HLLAPI emulator</p> <pre>Readtext ?result "X" "Y" "Z"</pre> <p>X = The number of characters to read.</p> <p>Y= The row from which the characters are read.</p> <p>Z= The column from which the characters are read.</p>
Example 2	<p>Windows script</p> <pre>ReadText #1004 ?result</pre>
Syntax examples	<pre>ReadText #301 ?Text ReadText ?Text 4 6</pre>

Example 1**Windows Application Definition**

The same Title and Class appear in the error message dialog box when a user fails to log in.

This example distinguishes between errors and provides users with more specific information, rather than a general message stating that their username and password is incorrect, or the account is locked. In this case, the example reads the error message, clicks *OK*, and prompts the user with a customized message.

```
# Log on Failed Message
Dialog
  Class #32770
  Title "Log on Failed"
EndDialog

ReadText #65535 ?ErrorMsg
Click #1
If "Invalid Username" -In ?ErrorMsg  DisplayVariables
  "Please verify your Username and try again." $Username
  Type $Username #1001
  Type $Password #1002
  Click #1
EndIf
If "Invalid Password" -In ?ErrorMsg  DisplayVariables
  "Please verify your Password and try again." $Password
  Type $Username #1001
  Type $Password #1002
  Click #1
EndIf
If "Account Locked" -In ?ErrorMsg  MessageBox "Your
account is locked. Please contact the Helpdesk on
x3849."
Endscript
EndIf
```

Example 2**Windows Application Definition**

This example reads the text from a Control ID and sets the database variable so the user is not prompted to set the variable.

```
# Log on Dialog Box
Dialog
  Class #32770
  Title "Log on"
EndDialog
ReadText #15 ?Database
If -Exists $Database
Else
  Set $Database ?Database
EndIf
Type $Username #1001
Type $Password #1002
Type $Database #1003
Click #1
```

Example 3	Terminal Launcher Application Definition
	This example reads a message in a Terminal Emulator and displays the message in a user-friendly format.
	<pre>ReadText ?Message 30 24 2 MessageBox ?Message</pre>

5.2.60 RegSplit

Use With	All
SecureLogin Version	3.5 to 6.1 SP1
Type	Action
Usage	RegSplit <Regex> <Input-String> [<Output-String1> [<Output-String2>]...]
Arguments	<p><Regex></p> <p>The regular expression.</p> <p><Input-String></p> <p>The string that to split.</p> <p><Output-String1></p> <p>The first subexpression.</p> <p><Output-String2></p> <p>The second subexpression.</p>
Description	<p>Use the <code>RegSplit</code> command to split a string by using a regular expression. <code><Output-String1></code> and <code><Output-String2></code> contain the first and second subexpressions.</p> <p>For more information on regular expression, see the Electronic Text Center (http://etext.lib.virginia.edu/services/helpsheets/unix/regex.html) or search the Microsoft MSDN library. (http://msdn.microsoft.com/en-us/library/default.aspx)</p>

Example	<p>Windows Application Definition</p> <p>This example copies text from Control ID #301 to the ?Text variable. The RegSplit command is then used to strip the username details out of the text that was read. The platform is set to that username, and the correct password is entered by SecureLogin.</p> <pre># Log on Dialog Box Dialog Class #32770 Title "Log on" EndDialog ReadText #65535 ?Text RegSplit "Please enter the password for (.) account" ?Text ?UserSetPlat ?User Type \$Username #1001 Type \$Password #1002 Click #1</pre>
Open Text Example	<pre>##?InputString: "This is a long string with a few components in it"</pre>
Command	<pre>RegSplit "This (.) a long (.) with (.) components (.)" ?InputString ?First ?Second ?Third ?Fourth</pre>
Result	<pre>?First = "is", ?Second = "string", ?Third = "a few", ?Fourth = "in it"</pre>

5.2.61 ReLoadPlat

When an application first presents a login screen, SecureLogin displays a message box prompting the user to select an appropriate platform from a list. After they are selected, SecureLogin enters the chosen platform's credentials into the application and submits them.

If login fails because of incorrect credentials, SecureLogin prompts the user to change his or her credentials. SecureLogin does not retain the platform details and prompts the user to reenter the information. This could result in the user changing the wrong credentials if they select the incorrect platform.

The SetPlat, ReLoadPlat, and ClearPlat commands resolve this issue. ReLoadPlat sets the current platform to the one that was last chosen (for the given application), or if a platform not previously selected, the command leaves it unset.

See also [Section 5.2.61, "ReLoadPlat," on page 132](#) and [Section 5.2.11, "ClearPlat," on page 71](#).

Use With	Startup, Terminal Launcher, Web, or Windows
SecureLogin Version	3.5 to 6.1 SP1
Type	Action

Usage	<p>Use the <code>ReLoadPlat</code> command at:</p> <ul style="list-style-type: none"> ◆ Login: Before the user first logs onto the application, call <code>ReLoadPlat</code>. This prevents the user from having to reselect a platform after a failed login. ◆ Failer Login: Call <code>ReLoadPlat</code> to reselect the platform that contained the incorrect credentials. This gives the user an opportunity to change the credentials by using a <code>ChangePassword</code> or a <code>DisplayVariables</code> command.
Arguments	None
Description	Use this command to set the current platform to the last one chosen by the application definition, or if a platform is not chosen, leaves the platform unset.
Example	<p>Windows Application Definition</p> <pre># ==== BeginSection: Application startup ==== Dialog Class "#32770" Title "Password Test Application" EndDialog ClearPlat # ==== EndSection: Application startup ==== # ==== BeginSection: Log on ==== Dialog Class "#32770" Title "Log on" Ctrl #1001 EndDialog ReLoadPlat SetPrompt "Username =====>" >Type \$Username #1001 SetPrompt "Password =====>" >Type \$Password #1002 SetPrompt "Domain =====>" >Type \$Domain #1003 Click #1 # ==== EndSection: Log on ==== ## ==== BeginSection: Log on Successful ==== Dialog Class "#32770" Title "Log on Successful" EndDialog ClearPlat Click #2 # ==== EndSection: Log on Successful =====</pre>

```

Example (Cont.)      # ==== BeginSection: Log on Failure ====
                    Dialog
                      Class "#32770"
                      Title "Log on Failure"
                    EndDialog
                    Click #2
                    ReLoadPlat
                    OnException ChangePasswordCancelled Call
                    ChangeCancelled
                    ChangePassword $password
                    ClearException ChangePasswordCancelled
                    Type -raw \Alt+F
                    Type -raw L
                    # ==== EndSection: Log on Failure ====

                    # ==== BeginSection: Change Credentials Cancelled ====
                    Sub ChangeCancelled
                      ClearPlat
                      EndScriptEndSub
                    # ==== EndSection: Change Credentials
                    Cancelled ===

```

5.2.62 Repeat/EndRepeat

Use With	All
SecureLogin Version	3.5 to 6.1 SP1
Type	Action
Usage	Repeat [Loop#] EndRepeat
Arguments	[Loop#]
	The number of times the repeat application definition block is repeated. If not specified, the repeat continues indefinitely unless broken by other commands.
Description	Use the Repeat command to establish an application definition block similar to the If command. The repeat block is terminated by an EndRepeat command. Alternatively, you can use the Break or EndScript commands to break out of the loop.
Syntax Examples	Repeat Repeat 3

Example**Terminal Application Definition**

This example uses the `Repeat` command to watch the screen for the messages and responds accordingly. You can use the `Break` command to jump to the next repeat loop in the application definition.

```
# Initial System Log on
WaitForText "login:"
Type $Username
Type @E
WaitForText "password:"
Type $Password
Type @E
Delay 500
#Repeat loop for error handling
Repeat
#Check to see if password has expired
If -Text "EMS: The password has expired."
    ChangePassword
    #Password
    Type $Password
    Type @E
    Type $Password
    Type @E
EndIf
#User has an invalid Username and / or # Password
stored.
If -Text "Log on Failed"
    DisplayVariables "The username and / or password
stored by SecureLogin is invalid. Please verify your
credentials and try again. IT x453."
    Type $Username
    Type @E
    Delay 500
    WaitForText "password:"
    Type $Password
    Type @E
    Delay 500
EndIf
# Account is locked for some reason, possibly inactive.
```

Example (Cont.)

```
    If -Text "Account Locked"
        MessageBox "Your account has been locked, possibly
due to inactivity for 40 days. Please contact the
administrator on x453."
    EndIf
# Main Menu, user has logged on successfully.
    If -Text "Application Selection"
        Break
    EndIf
Delay 100
EndRepeat
```

5.2.63 RestrictVariable

Use With**All**

SecureLogin Version	3.5 to 6.1 SP1
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 for the variable.
Description	<p>Use the <code>RestrictVariable</code> command to monitor a <Variable> and enforce a specified <Password-Policy> on the <Variable>. Any variable specified must match the policy or it is not saved.</p> <p>When restricting variables by using policies, if you are using a more restrictive policy than is already in place, and you restrict a variable that does not match the policy now in effect, then the user cannot save it the first time. This is because when SecureLogin detects that there is no saved credential, a user who has a password of six characters, cannot save it if the policy restricts the <code>\$Password</code> variable to eight characters and two numbers.</p> <p>“Example 2” on page 138 works around this by restricting a new password variable (<code>?NewPwd</code>), instead of restricting the <code>\$Password</code> variable. The user can store an existing password when SecureLogin prompts for the credentials the first time, and enforces the stronger password policy when the password expires in x days.</p> <p>You can restrict any variable by using a password policy, not just a <code>\$Password</code>. You can also use <code>RestrictVariable</code> to make sure other variables are entered in the correct format. For example, you can enforce that <code>\$Username</code> is always lowercase or <code>\$Database</code> is 6 characters and no numbers.</p>

Example 1**Windows Application Definition**

This example uses the application definition to restrict the \$Password variable to the Finance password policy. The user's password must match the policy when he or she first saves the credentials. When the password requires changing, the application definition generates a new password randomly based on that policy (no user intervention is required).

```
# Set the Password to use the Finance Password Policy
RestrictVariable $Password FinancePwdPolicy
```

```
#Log on Dialog Box
Dialog
  Class #32770
  Title "Log on"
EndDialog
```

```
Type $Username #1001
Type $Password #1002
```

```
#Change Password Dialog Box
```

```
Dialog
  Class #32770
  Title "Change Password"
EndDialog
  Type $Username #1015
  Type $Password #1004
  ChangePassword $Password Random
  Type $Password #1005
  Type $Password #1006
  Click #1
```

Example 2**Windows Application Definition**

This example uses the application definition to restrict the ?NewPwd variable to the Finance password policy. When the application starts for the first time and prompts the user to enter his or her credentials, then the current password (\$Password) is saved and used.

When the password expires, the password policy is enforced on any new password. This is a way to enforce more restrictive password policies than are currently in place when you cannot guarantee that all existing passwords meet the new policy.

```
# Set the Password to use the Finance Password Policy
RestrictVariable ?NewPwd FinancePwdPolicy
# Log on Dialog Box
Dialog
  Class #32770
  Title "Log on"
EndDialog
  Type $Username #1001
  Type $Password #1002
Click #1
# Change Password
Dialog Box
Dialog
  Class #32770
  Title "Change Password"
EndDialog
  Type $Username #1015
  Type $Password #1004
  ChangePassword ?NewPwd Random
  Type ?NewPwd #1005
  Type ?NewPwd #1006
  Set $Password ?NewPwd
Click #1
```

5.2.64 Run

Use With	Startup, Terminal Launcher, Web, or Windows
SecureLogin Version	3.5 to 6.1 SP1
Type	Action
Usage	Run <Command> [<Arg1> [<Arg2>] ...]
Arguments	<Command> The full path of the program to execute. <Arg1>, <Arg2> An optional list of arguments and switches for the command.
Description	Use the Run command to launch the program specified in <Command> with the specified optional [<Arg1> [<Arg2>] ...] arguments. The application definition does not wait for the launched program to complete.

Example	<p>Startup</p> <p>This example prompts the user to start the Finance System.</p> <p>If they click:</p> <ul style="list-style-type: none"> ◆ Yes, the <code>Run</code> command is used to start the application with the necessary switches. ◆ No, a message box is displayed, and the application is not started. <pre> MessageBox "Would you like to connect to the Finance System?" -YesNo ?Result If ?Result Eq "Yes" Run "C:\Program Files\HRS\Finance.exe" "/DB:HRS" "/" Debug" Else MessageBox "You have chosen not to run the FinanceSystem. Please do so manually." EndScript EndIf </pre>
---------	--

5.2.65 Select

Use With	Java and Advanced Web application definitions.
SecureLogin Version	Introduced in version 6.1.000
Type	Action
Usage	Select <Text of Item to select> [<#Item Number>]
Arguments	<p><Text of Item to select></p> <p>The text item that you want SecureLogin to select in the list box.</p> <p><#Item Number></p> <p>When multiple list boxes are found, this specifies which list box to address.</p>
Description	Use the <code>Select</code> command to select entries from a combo/list style control.
Examples	<p>This example picks an item from the session list/comb :</p> <pre>Select ?session #1</pre> <p>This example selects a tab within another tab control. When one tab control is contained within another, the tab selection order is irrelevant.</p> <pre>Select "Quick Connect" #70 Select "Connection" #69</pre> <p>This example selects a cell from within a table:</p> <pre>Select "[0,0]" #1 If -text "User" #1 Select "[0,1]" #1 Type \$Username #1 Endif</pre>

5.2.66 SelectListBoxItem

Use With	Advanced Web application definitions
SecureLogin Version	3.5 to 6.1 SP1
Type	Action
Usage	SelectListBoxItem <Text of Item to set to> [<#Item Number>] [<-multiselect>]
Arguments	<p><Text of Item to set to></p> <p>The text item that you want SecureLogin to select in the list box.</p> <p><#Item Number></p> <p>When multiple list boxes are found, this specifies which list box to address.</p> <p><-multiselect></p> <p>Used to select multiple list box entries by using a subsequent SelectListBoxItem command.</p>
Description	<p>Use the SelectListBoxItem command to select entries from a list box.</p> <p>For instruction on determining item numbers, see Section 5.2.22, "DumpPage," on page 84.</p>
Example	<pre>SelectListBoxItem "Remember Defects" #2 -multiselect SelectListBoxItem "Remember Enhancements" #2 - multiselect</pre>

5.2.67 SelectOption

Use With	Advanced Web application definitions created by using the Web Wizard.
SecureLogin Version	3.6.1.0 to 6.1 SP1
Type	Action
Usage	SelectOption <#FormID:FieldID:OptionID -select "select" #FormID:FieldID -clear>
Arguments	<p>SelectOption</p> <p>Used to select or deselect options within a list box or combo box.</p> <p>-select "select"</p> <p>Selects or deselects a specific option.</p> <p>"select" is a Boolean value, either "true" or "false".</p> <p>-clear</p> <p>Deselects all options for the given control.</p>

Description	Use the <code>SelectOption</code> command to select or deselect options within a list box or combo dialog box.
Example	<p>This example selects the Default User and sets a new platform so that only the default user can log in to the application. In this case, <code>SetPlat</code> creates a new platform called Default and the respective <code>\$Username</code> and <code>\$Password</code> are saved there.</p> <pre># log on Dialog Box Dialog Class #32770 Title "Log on" EndDialog SelectOption "Default User" SetPlat ?Default Type \$Username #1001 Type \$Password #1002 Click #3</pre>

5.2.68 SendKey

Use With	Terminal Launcher
SecureLogin Version	3.5 to 6.1 SP1
Type	Action
Usage	<code>SendKey <Text></code>
Arguments	<code><Text></code>
Description	<p>The text typed into the emulator screen.</p> <p>Use the <code>SendKey</code> command to work only with Generic and Advanced Generic emulators. You can use the <code>SendKey</code> command in the same manner as the <code>Type</code> command. Generally, the <code>Type</code> command is the preferred command to use. The <code>Type</code> command places the text into the clipboard, and then pastes it into the emulator screen. The <code>SendKey</code> command enters the text directly into the emulator screen.</p> <p>Variables do not work with the <code>SendKey</code> command. If you want to use variables, use the <code>Type</code> command.</p> <p>The <code>Type</code> command has many special functions, and some you can use with the <code>SendKey</code> command. For more information, see Section 5.2.87, "Type," on page 162, and for more details on these functions, see Chapter 7, "Reference Commands and Keys," on page 177.</p>

Example	<p>Terminal Launcher Application Definition</p> <p>The example sends the username and password to the terminal emulator.</p> <pre>#Send Username SendKey "writer" SendKey "\N" #Send Password SendKey "Hu7%f" SendKey "\N"</pre>
---------	--

5.2.69 Set

Use With	All
SecureLogin Version	3.5 to 6.1 SP1
Type	Action
Usage	Set <Variable> <Data>
Arguments	<p><Variable></p> <p>The variable to which the data is being assigned.</p> <p><Data></p> <p>The text or variable read from and assigned to the variable.</p>
Description	Use the <code>Set</code> command to copy the value of <Data> into <Variable>. The <Data> can be any text, or another variable, but the <Variable> must be either a ?Variable or \$Variable.
Example 1	<p>Windows Application Definition</p> <p>This example uses the application definition to set a ?RunCount variable to count the number of times the application is run.</p> <pre># Log on Dialog Box Dialog Class #32770 Title "Log on" EndDialog If ?RunCount Eq <NOTSET> Set ?RunCount "1" Else Increment ?RunCount EndIf Type \$Username #1001 Type \$Password #1002 Click #1</pre>

Example 2**Windows Application Definition**

This example uses the application definition to set the ?NewPwd to the stored \$Password variable.

```
# Log on Dialog Box
Dialog
  Class #32770
  Title "Log on"
EndDialog

  Type $Username #1001
  Type $Password #1002
Click #1

# Change Password Dialog Box
Dialog
  Class #32770
  Title "Change Password"
EndDialog

  Type $Username #1015
  Type $Password #1004
  ChangePassword ?NewPwd Random
  Type ?NewPwd #1005
  Type ?NewPwd #1006
  Set $Password ?NewPwd
Click #1
```

Example 3**Windows Application Definition**

This example uses the application definition to read the value of Ctrl #15, and sets the \$Database variable so the user does not need to set the variable.

```
# Log on Dialog Box
Dialog
  Class #32770
  Title "Log on"
EndDialog

ReadText #15 ?Database
If -Exists $Database
Else
  Set $Database ?Database
EndIf
```

5.2.70 SetCheckBox

Use With	Advanced Web Application Definition
SecureLogin Version	3.5 to 6.1 SP1
Type	Action
Usage	SetCheckBox <Item Number> <Option>

Arguments	<p><Item Number></p> <p>The check box in reference to the number of check boxes found.</p> <p><Option></p> <p>Specifies the status of the check box as Checked or Unchecked.</p>
Description	Use the <code>SetCheckBox</code> command to select or clear a check box.
Example	<pre> Messagebox "Scroll down so you can see the 'Search Language' section and all the Languages with the check boxes then click OK on this messagebox"setcheckbox #1 "checked" setcheckbox #2 "checked" setcheckbox #3 "checked" setcheckbox #4 "checked" setcheckbox #25 "checked" setcheckbox #26 "checked" setcheckbox #27 "checked" Messagebox "Did it select the first four languages and Norwegian, Polish and Portuguese Languages" -yesno ?advweb if ?advweb eq yes set ?cmd37 "Setcheckbox command worked"else set ?cmd37 "Setcheckbox failed" endifset checkbox #1 "unchecked" setcheckbox #2 "unchecked" setcheckbox #3 "unchecked" setcheckbox #4 "unchecked" setcheckbox #26 "unchecked" setcheckbox #27 "unchecked" Messagebox "Did it clear all the languages except Norwegian" -yesno ? advweb2 if ?advweb2 eq yes set ?cmd38 "setcheckbox command worked" else set ?cmd38 "setcheckbox failed" endif </pre>

5.2.71 SetCursor

Use With	Terminal Launcher (Only available in HLLAPI and some DDE emulators)
SecureLogin Version	3.5 to 6.1 SP1
Type	Action
Usage 1	SetCursor <Screen-Position>
Usage 2	SetCursor <X Co-ordinate> <Y Co-ordinate>

Arguments	<p><Screen-Position></p> <p>The position on the screen to move the cursor.</p> <p><X Co-ordinate></p> <p>The horizontal coordinate. When specified, a row or column conversion is carried out before the cursor is set to the position.</p> <p><Y Co-ordinate></p> <p>The vertical coordinates. When specified, a row or column conversion is carried out before the cursor is set to the position.</p>
Description	<p>Use the <code>SetCursor</code> command to set the cursor to a specified <ScreenPosition> or <X Co-ordinate> <Y Co-ordinate>.</p> <p>The position is noted by a number greater than 0 (zero), for example, <code>SetCursor 200</code>. Terminal Launcher displays an error message if the screen position is invalid.</p>
Syntax examples	<pre>SetCursor 200 SetCursor 100 500</pre>
Example	<p>Terminal Launcher Application Definition</p> <p>This example sets the cursor to the correct position, and then you enter credentials.</p> <pre>SetCursor 200 Type \$Username Type @E Type \$Password Type @E</pre>

5.2.72 SetFocus

Use With	Java, Web, Windows
SecureLogin Version	3.5 to 6.5
Type	Action
Arguments	<p><#Ctrl-ID></p> <p>The ID number of the control to which the keyboard focus is directed.</p>
Description	<p>Use the <code>SetFocus</code> command to set the keyboard focus to a specified <#Ctrl-ID>.</p> <p>A valid <#Ctrl-ID> is required for the <code>SetFocus</code> command to function correctly.</p>

Example	<p>Windows Application Definition</p> <p>This example sets the focus to the username field (#1001). The username is typed and a tab stop is simulated, and then the password is typed and pressing ENTER is simulated.</p> <pre> # Log on Dialog Box Dialog Class #32770 Title "Log on" EndDialog SetFocus #1001 Type \$Username Type \T Type \$Password Type \N </pre>
---------	--

5.2.73 SetPlat

Use With	All
SecureLogin Version	3.5 to 6.1 SP1
Type	Action
Usage 1	SetPlat <Application-Name>
Usage 2	SetPlat <RegEx> <Variable> <#Ctrl-ID>
Arguments	<p><Application-Name></p> <p>Application name from which to read the variables.</p> <p><RegEx></p> <p>Regular expression to use as application name.</p> <p><Variable></p> <p>Use a previously set ?Variable, such as <code>PickList</code> (see Section 5.2.55, "PickListAdd," on page 123).</p> <p><#Ctrl-ID></p> <p>The control ID number of the regular expression.</p>

NOTE: For information regarding regular expressions, see [Electronic Text Center. \(http://etext.lib.virginia.edu/services/helpsheets/unix/regex.html\)](http://etext.lib.virginia.edu/services/helpsheets/unix/regex.html)

Description

By default, variables are stored directly against the platform or application on which you have SecureLogin enabled. For example, if you enable `Groupwise.exe`, the Groupwise credentials are stored against the `Groupwise.exe` platform.

`SetPlat` sets the platform or application from which variables are read and saved if you have:

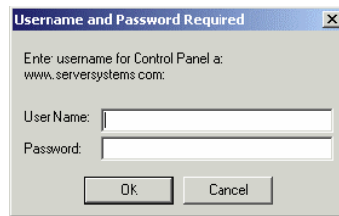
- ◆ Multiple accounts (for example, your own login and an admin login) accessing the same platform or application.
- ◆ Multiple platforms or applications using a common set of credentials?

Other uses of `SetPlat` include:

- ◆ Configuring application1 to read its `$Username` and `$Password` from application2. This saves a user from entering the credentials twice and needing to remember to update them in both locations when they change, and so on.
 - ◆ Configuring application1, application2, and application3 to read the user's credentials from Platform Common. This results in a single store of common credentials that you only need to update once.
-

Example 1**Web Application Definition**

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



When you specify the *Title*, *Class*, *Username*, and *Password* fields for this dialog box, they are always the same. If you stored the *Username* and *Password* against this platform without using the `SetPlat` command, the same *Username* and *Password* for `www.serversystems.com` are entered to log in to any site (and are obviously invalid for any other site).

However, the previous dialog box always contains the name of the Web site to which to log in. You can use this name as the unique identifier in order to set a new platform and to save the login credentials.

Using a dialog block with a `SetPlat` statement: The solution is to use a dialog block with a `SetPlat` statement, such as:

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

The power of this application definition is the line:

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

This reads the line from dialog control ID 331, enters the username for Control Panel at `www.serversystems.com`Next, and applies the regular expression to this text. Regular expressions are a way of manipulating text strings; however, for most purposes a few very basic commands work.

For information regarding regular expressions, see [Electronic Text Center](http://etext.lib.virginia.edu/services/helpsheets/unix/regex.html). (<http://etext.lib.virginia.edu/services/helpsheets/unix/regex.html>)

When the user runs the application definition, he or she sees the username and password saved as `www.serversystems.com`. The text matched inside the brackets then becomes the symbol application. If a dialog `<#Ctrl-ID>` is not specified, the symbol application is unconditionally changed to the application specified in `<RegEx>`. An unconditional `SetPlat` command is only valid if specified before `Dialog/EndDialog` statements.

Example 2**Windows Application Definition**

This example displays a pick list and sets a new platform so multiple users can log in to the application. In this case, `SetPlat` creates a new platform called Default User, Global Administrator, or Regional Administrator, and the respective `$Username` and `$Password` is saved there.

```
# log on Dialog Box
Dialog
  Class #32770
  Title "Log on"
EndDialog

PickListAdd "Default User"
PickListAdd "Global Administrator"
PickListAdd "Regional Administrator"
PickListDisplay ?Choice "Please select the account you
wish to use"-NoEdit
SetPlat ?Choice
Type $Username #1001
Type $Password #1002
Click #3
```

5.2.74 SetPrompt

Use with	All
SecureLogin Version	3.5 to 6.1 SP1
Type	Action
Usage	SetPrompt <Prompt-Text>
Arguments	<Prompt-Text>
	The customized text prompt displayed in the Enter SecureLogin Variables dialog box.
Description	Use the <code>SetPrompt</code> command to customize the text in the Enter SecureLogin Variables dialog boxes. These dialog boxes are used to prompt the user for new variables. You can also use the <code>DisplayVariables</code> command to customize the prompt text in the dialog box (for previously stored variables).
	For more information, see Section 5.2.20, "DisplayVariables," on page 82 .
	NOTE: Positioning of the <code>Setprompt</code> command is crucial. Position it before the first usage of each variable to name that variable, and apply the final <code>Setprompt</code> to the text displayed at the top of the prompt screen.

Example 1**Windows Application Definition**

This example replaces the default text prompt in the Enter SecureLogin Variables dialog box, and places the `SetPrompt` command at the bottom of the application definition.

```
# Log on Dialog Box
Dialog
Class #32770
Title "Log on"
EndDialog
Type $Username #1001
Type $Password #1002
Click #1
SetPrompt "Please enter your Username and Password for
accessing the Human Resources system. These credentials
will be remembered by SecureLogin and you will be
automatically logged on in future. IT Helpdesk x4564"
```

Example 2**Windows Application Definition**

This example replaces the text prompt next to any variable entry field in the Enter SecureLogin Variables box, and places the `SetPrompt` command immediately before the variable in the application definition.

```
# Log on Dialog Box
Dialog
Class #32770
Title "Log on"
EndDialog
SetPrompt "Enter Username==>"
Type $Username #1001
SetPrompt "Enter Password==>"
Type $Password #1002
Click #1
SetPrompt "Please enter your Username and Password for
accessing the Human Resources system. These credentials
will be remembered by SecureLogin and you will be
automatically logged on in future. IT Helpdesk x4564"
```

5.2.75 -SiteDeparted

Use With	Web
Novell SecureLogin version	3.5 or later
Type	Action
Argument	<code>SiteDeparted</code> is a conditional variable.
Description	Use the <code>SiteDeparted</code> variable in Web scripts to see if the current document is still active when used as part of an If statement.

Example	<p>The following example checks if the user has navigated away from the current Web site or not.</p> <p>If the users have navigated away from the Website, it informs the users and exists the script.</p> <pre> If -SiteDeparted MessageBox "Script terminated, we have left the web- site" EndScript EndIf </pre>
---------	---

5.2.76 Site/Endsite

Use With	Advanced Web application definitions created by using the Web Wizard
SecureLogin Version	3.6.1 to 6.1 SP1
Type	Action
Usage	Site ["Name" [-userid "userid"] [-initial -subsequent -recent timeout] [-nonexclusive]]

Arguments	<p>Site</p> <p>The <code>Site/EndSite</code> commands are used to match a particular site given a set of filters. <code>Site/Endsite</code> usage is much the same as the <code>Dialog/EndDialog</code> commands found in the windows scripting commands.</p> <p>"Name"</p> <p>Name is a static string used to denote the site being matched. The Name cannot be a variable and the same value can be used by multiple site commands to specify a match for the same site under differing conditions.</p> <p><code>-userid "userid"</code></p> <p>Specifies the default set of credentials to be used for this site block.</p> <hr/> <p>NOTE: "userid" must be a static string.</p> <hr/> <p><code>-initial</code></p> <p>Specifies that this site block only matches the first time.</p> <p><code>-subsequent</code></p> <p>Specifies that this site block only matches after an initial match has already been made.</p> <p><code>-recent timeout</code></p> <p>Specifies that this site block only matches if a previous match was made within the given timeout period.</p> <p>The timeout is given in milliseconds.</p> <p><code>-nonexclusive</code></p> <p>Specifies that even if this site block matches, other scripts and wizards are not prevented from running.</p>
Description	<p><code>Site/EndSite</code> begins and ends an Application Definition, in place of <code>Dialog/EndDialog</code>.</p> <p>The <code>Site/EndSite</code> commands have been added to allow for much finer control of Web site matching. No longer is a URL all that can be matched on. Detailed information of the loaded Web site can now be matched upon and used to execute blocks of scripting commands.</p> <p><code>Site/EndSite</code> blocks are used to define all the parameters SecureLogin expects to find on a Web page to run the application definition.</p> <p><code>Match</code> commands can be used to filter a given site. If one of the contained match commands fails to match, then the site block fails to match as a whole.</p>
Example 1:	<p>This simple example would locate the Web site <code>www.mybank.com</code>.</p> <pre># === My Bank Initial Login === Site "www.mybank.com" -userid "My Login Credentials" -initial EndSite</pre>

Example 2:

This simple example would locate the Web site `www.google.com`, locate the login form and login to the user's account using the users e-mail address, account number, and password.

```
# === Login Application Definition #2 ==
# === Google Initial Login ===
#=====
Site Login -userid "Google Log On" -initial
MatchDomain "www.google.com"
MatchField #1:1 -name "Email" -type "text"
MatchField #1:2 -name "Passwd" -type "password"
MatchField #1:3 -name "Cookie" -type "check"
EndSite
SetPrompt "Enter your user credentials"
TextInput #1:1 -value "$Username"
TextInput #1:2 -value "$Password"
FocusInput#1:2 -focus "true"
BooleanInput #1:3 -check "false"
PressInput
Endscript
```

5.2.77 StrCat

Use With	All
SecureLogin Version	3.5 to 6.1 SP1
Type	Action
Usage	StrCat <Variable> <Input-String1> <Input-String2>
Arguments	<Variable> The variable to which you want to result saved. <Input-String1> First data string or variable. <Input-String2> Second data string or variable.
Description	Use the <code>StrCat</code> command to append the second data string to the first data string. For example, <code>StrCat ?Result "SecureRemote " "\$Username"</code> . In this case <code>"\$Username"</code> is "Tim", and the variable <code>"?Result"</code> now contains the value <code>"SecureRemote Tim"</code> .

Example:

Windows Application Definition

This example reads the username from #1001 into ?Username and uses the StrCat command to join the username to the password. The result is a LogonID, which SecureLogin uses to log in to the system.

```
# Log on Dialog Box
Dialog
  Class #32770
  Title "Log on"
EndDialog

ReadText #1001 ?Username
StrCat ?LoginID ?Username $Password
Type ?LoginID #1002
Click #1
```

5.2.78 StrLength

Use With All

SecureLogin Version 3.0.4 to 6.1 SP1

Type Variable Manipulator

Usage StrLength <Destination> <String>

Arguments <Destination>

The output variable. Also the input variable if no source is specified.

<String>

The string whose length you want to measure.

Description Use the StrLength command to count the number of characters in a variable and output that value to the destination variable.

Example Windows Application Definition

This example reads the password from #301 and then uses StrLength to count the number of characters. If it is less than 4, an error message is displayed.

```
# Log on Dialog Box
Dialog
  Class #32770
  Title "Log on"
EndDialog

ReadText #301 ?Password
StrLength ?Length ?Password
If ?Length Lt "4"
  MessageBox "Password is too short"
EndIf
```

5.2.79 StrLower

Use with	All
SecureLogin Version	3.0.4 to 6.1 SP1
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 a variable is not specified, SecureLogin reads the destination variable, makes the necessary changes, and writes over the variable.
Description	Use the StrLower command to modify a variable so that all the characters are lowercase. If only a: <ul style="list-style-type: none">♦ If only a destination variable is specified, the string is read from the destination, then is stored back to it.♦ If only 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	Windows Application Definition The example reads the username from #1001 and copies it into ?Username. The StrLower command is then used to make sure the username is all lowercase. <pre># Log on Dialog Box Dialog Class #32770 Title "Log on" EndDialog ReadText #1001 ?Username StrLower ?LowerCaseUsername ?Username Type ?LowerCaseUsername #1002 Click #1</pre>

5.2.80 StrUpper

Use With	All
SecureLogin Version	3.0.4 to 6.1 SP1
Type	Variable Manipulator

Arguments	<p><Destination></p> <p>The output variable. Also the input variable if no source is specified.</p> <p>[<Source>]</p> <p>The input variable. If a variable is not specified, SecureLogin reads the destination variable, makes the necessary changes, and writes over the variable.</p>
Description	<p>Use the <code>StrUpper</code> command to modify a variable so that all the characters are uppercase.</p> <p>If only a:</p> <ul style="list-style-type: none"> ◆ If only a destination variable is specified, the string is read from the destination and is then stored back to it. ◆ If only 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	<p>Windows Application Definition</p> <p>This example reads the username from #1001 and copies it into ?Username. The <code>StrUpper</code> command is then used to make sure the username is all uppercase.</p> <pre># Log on Dialog Box Dialog Class #32770 Title "Log on" EndDialog ReadText #1001 ?Username StrUpper ?UpperCaseUsername ?Username Type ?UpperCaseUsername #1002 Click #1</pre>

5.2.81 Sub/EndSub

Use With	Startup, Terminal Launcher, Web, or Windows
SecureLogin Version	3.5 to 6.1 SP1
Type	Flow Control
Usage	Sub <Name> EndSub
Arguments	<p><Name></p> <p>Any name entered to identify the subroutine.</p>
Description	<p>Use the <code>Sub/EndSub</code> commands around a block of lines within an application definition to denote a subroutine.</p> <p>You can also call a subroutine by using the <code>Call</code> command. For more information, see Section 5.2.8, "Call," on page 68.</p>

Example**Terminal Launcher Application Definition**

This example checks the emulator screen for the text login or wrong password. If either is found, the appropriate subroutine is called and run before the next part of the application definition.

```
If -Text "Log on"  
    Call "Log on"  
EndIf  
If -Text "Wrong Password"  
    Call "WrongPassword"  
EndIf  
Sub Login  
    Type $Username  
    Type @E  
    Type $Password  
    Type @E  
EndSub  
Sub WrongPassword  
    DisplayVariables "Enter correct password"  
    $Password  
    Call Login  
EndSub
```

5.2.82 Submit

Use With	Web
SecureLogin Version	3.5 to 6.1 SP1
Type	Action
Usage	Submit
Arguments	None

Description	<p>Use the <code>Submit</code> command only in Web application definitions, and only with Internet Explorer, to allow for enhanced control of how and when a form is submitted. The <code>Submit</code> command performs a Submit on the form in which the first password field is found. The <code>Submit</code> command is ignored if it is used with Netscape.</p> <p>The function performed by the <code>Submit</code> command is automatically performed by Web application definition by default. For example, the application definition:</p> <pre>Type \$Username Type \$Password Password</pre> <p>Types the username and password and submits the form.</p> <p>However, submits do not occur automatically if any of the following commands are in the application definition: <code>Type \N</code>, <code>Type \T</code>, <code>Submit</code>, or <code>Click</code>. If any of these commands are used, you must use the <code>Submit</code> command or some other means to submit the form.</p> <p>Furthermore, an automatic submit does not occur if you type text into a specific text entry field. For example, in the application definition segment below, the <code>Submit</code> command must follow the <code>Type</code> command for the application definition to work properly:</p> <pre>Type \$Username #1001 Submit</pre>
Example.	<p>Web Application Definition</p> <p>This example enters the username and password and then executes a manual <code>Submit</code>.</p> <pre>Type \$Username #1 Type \$Password #2 Submit</pre>

5.2.83 Subtract

Use With	Startup, Terminal Launcher, Web, or Windows
SecureLogin Version	3.0 to 6.1 SP1
Type	Variable Manipulator
Usage	Subtract <Start-Value> <Subtract-Value> [?Result]

Arguments	<p><Start-Value></p> <p>The <Start-Value> argument is the start number from which the second argument is subtracted. This argument contains the result if the optional [?Result] argument is not passed in.</p> <p>If used:</p> <ul style="list-style-type: none"> ◆ If used without the [?Result] argument, then <Start-Value> must be a SecureLogin variable, for example, ?StartValue or \$StartValue. ◆ If used with the [?Result] argument, then <Start-Value> can be a SecureLogin variable or a numeric value. <p><Subtract-Value></p> <p>The <Subtract-Value> argument is the number subtracted from the first argument. <Subtract-Value> can be a SecureLogin variable or a numeric value.</p> <p>[?Result]</p> <p>The result of the equation. This argument is optional, but If used, set to <Start-Value> - <Subtract-Value>. The [?Result] must be a SecureLogin variable, for example, \$Result or ?Result.</p>
Description	<p>Use the <code>Subtract</code> command to subtract one value from another. This is useful if you are implementing periodic password change functionality for an application. You can use the <code>Subtract</code> command (in conjunction with the <code>Divide</code> function and the <code>Slna DLL</code>) to determine the number of days that have elapsed since the last password change. Other numeric commands include the <code>Add</code>, <code>Divide</code>, and <code>Multiply</code>.</p> <p>For more information see:</p> <ul style="list-style-type: none"> ◆ Section 5.2.2, "ADD," on page 62 ◆ Section 5.2.21, "Divide," on page 83 ◆ Section 5.2.52, "Multiply," on page 115 <hr/> <p>NOTE: The <code>Subtract</code> command correctly subtracts when <StartValue>, <Subtract-Value> and <Result-Value> are between -2147483648 and +2147483647.</p>
Syntax Examples:	<pre>Subtract "1" "2" ?Result Subtract ?LoginAttempts ?LoginFailures Subtract ?LoginAttempts ?LoginFailures ?Result Subtract ?LoginAttempts "3" Subtract ?LoginAttempts "3" ?Result</pre>
Example	<p>Windows Application Definition</p> <p>This example reads the values of Control IDs 103 and 104 into variables. From there they are subtracted, and typed into Control ID 1.</p> <pre>ReadText #103 ?Number1 ReadText #104 ?Number2 Subtract ?Number1 ?Number2 ?Result Type ?Result</pre>

5.2.84 Tag/EndTag

Use With	Advanced Web Application Definition
SecureLogin Version	3.5 to 6.1 SP1
Type	Tag Specifier
Usage	Tag EndTag
Arguments	None
Description	Use the <code>Tag/EndTag</code> commands to find HTML tags.
Example	This example finds the form that has an attribute of name with a value of log in. <pre>Tag "Form" Attribute "Name" "Log on"EndTag</pre>

5.2.85 TextInput

Use With	Advanced Web application definitions created by using the Web Wizard.
SecureLogin Version	3.6.1.0 to 6.1 SP1
Type	Action
Usage	<code>TextInput #FormID:FieldID -value "value"</code>
Arguments	<code>#FormID</code> The ID to be given to the matched form. The ID must be a static unsigned integer. <code>#FieldID</code> The ID to be given to the matched field. The ID must be a static unsigned integer. <code>-value "value"</code> The text value to be input.
Description	Used inside a site block to input text into a specified field.

Example In this example, the text value of the system username and password are passed to the application definition.

```
# === Login Application Definition #2 ==
# === Google Initial Login ====
#=====
Site Login -userid "Google Log On" -initial
MatchDomain "www.google.com"
MatchField #1:1 -name "Email" -type "text"
MatchField #1:2 -name "Passwd" -type "password"
MatchField #1:3 -name "Cookie" -type "check"
EndSite
SetPrompt "Enter your user credentials"
TextInput #1:1 -value "$Username"
TextInput #1:2 -value "$Password"
FocusInput#1:2 -focus "true"
BooleanInput #1:3 -check "false"
PressInput
Endscript
```

5.2.86 Title

Use With	Java, Windows
SecureLogin Version	3.5 to 6.1 SP1
Type	Dialog Specifier
Usage	Title <Window-Title>
Arguments	<Window-Title> The text to test against the window title.
Description	<p>Use the <code>Title</code> command to retrieve the title of a window and compare it against the string specified in the <Window-Title> argument. For this block of the application definition to run, the retrieved window title and the <Window-Title> argument must match the text supplied to the <code>Title</code> command in the dialog block.</p> <p><code>Title</code> is one of the main commands to identify a window. However, the <code>Title</code> command alone might not be enough. If there is more than one window in a platform (application) with the specified title, the SecureLogin application definition runs every time that window is detected.</p> <p>Always place the <code>Title</code> command after all other commands in the Dialog block.</p> <p>To uniquely identify a window, the <code>Title</code> command is typically used with the <code>Class</code> or <code>Ctrl</code> commands. For more information, see Section 5.2.10, "Class," on page 70 and Section 5.2.15, "Ctrl," on page 77.</p>

NOTE: Use the Window Finder tool to determine the window title.

Example	<p>Windows Application Definition</p> <p>This example tests the dialog box to see if it has the correct title. If the title is not correct, the application definition passes on to the next dialog block.</p> <pre># Log on Dialog Box Dialog Class #32770 Title "Logon" EndDialog Type \$Username #1001 Type \$Password #1002 Click #1</pre>
---------	--

5.2.87 Type

Use With	Java, Terminal Launcher, Web, or Windows
SecureLogin Version	3.5 to 6.1 SP1
Type	Action
Terminal Usage	Type [-Raw] <Text>
Windows Usage	Type <Text> [<#Ctrl-ID>] Type [-Raw] <Text> Type [-order] <Text> [<#Order-ID>] Type [-msg] <Text> [<#Ctrl-ID>]
Web Usage	Type <Text> [<#Field-ID>] Type <Text> ["password"]

Arguments**[-Raw]**

By default, when typing into a Terminal Emulator or Windows application, SecureLogin verifies that the window exists before continuing. This verification process is disabled when the -Raw argument is provided. Further, instead of trying to set the text in the field directly, this option simulates actual keystrokes, causing SecureLogin to type into whichever window has focus.

[-order]

If the control IDs are not constant, utilize the -order switch to type into a control based on the creation order and not the tab order.

[-msg]

This option can be used when a `Type` command is sending the data correctly, but the application is not successfully reading the data. The -msg modifier sends the data character by character versus sending the text string all at once. This -msg option is often useful for older windows applications, particularly older versions of Lotus* Notes*.

<Text>

The text to type into this area. This text can be static text, such as ABC, or any SecureLogin variable, such as `$Username`.

[<#Ctrl-ID>]

For Windows application definitions, this optional argument specifies the control into which to type the text. Use the Windows Finder tool to extract these control IDs. For more information, see [“Windows-Specific:” on page 164](#).

[<#Order-ID>]

For Windows application definitions, this parameter specifies the control (based on the creation order) into which to type the text.

[<#Field-ID>]

For Web application definitions, this optional argument specifies the text field into which to type the text. For more information, see [“Web-Specific:” on page 164](#).

[password]

For Web application definitions, this optional argument specifies to perform this type this the password field on this form. If [password] is used, that application's application definition cannot use a `<#Ctrl-ID>` argument. For more information, see [“Web-Specific:” on page 164](#).

Description

Use the `Type` command to enter data, such as usernames and passwords, into applications. There are reserved character sequences that are used to type special characters, for example TAB and ENTER. If it is not possible to determine Control IDs in a Windows application, and the `Type` command is not working, use the `SendKey` command instead.

Windows-Specific: In Windows, if the `<#Ctrl-ID>` argument is:

- ◆ If the `<#Ctrl-ID>` argument is provided, it must be a number that refers to a control ID as identified by the Windows Finder tool. SecureLogin then sends the contents of the `<Text>` argument directly to the window and to the specific control that matches the `<#Ctrl-ID>` argument.
- ◆ If the `<#Ctrl-ID>` argument is not specified, SecureLogin sends keystrokes to whichever control has focus. In the Windows environment, the `-Raw` option is often useful when the Window Finder tool is unable to determine control IDs for the text entry areas of an application, or these control IDs are changing. If you are using the `-Raw` option, then you cannot use the `<#Ctrl-ID>` argument.

Web-Specific: For Web pages there are two ways to specify which field receives `<Text>`.

- ◆ The first method uses absolute positioning by means of the `<#FieldID>` argument. The `<#Field-ID>` is a number that refers to the location of the field within the HTML form. For example, `#1` refers to the first text entry field in the Web form; `#2` refers to the second text entry field, and so on.
- ◆ The second method uses relative position using the password argument. In this method the SecureLogin agent first locates the text field within the HTML form that is a password field, and types `<Text>` into that field. Other `Type` commands send their `<Text>` parameters to fields that are relative to the first password field.

For example, the `Type` command immediately preceding the `Type` command that has the `[Password]` argument is sent to the text field immediately preceding the first password field.

Example 1

Windows Application Definition

This example is a typical use of the `Type` command in a Windows application definition.

```
# Log on Dialog Box
Dialog
  Class #32770
  Title "Log on"
EndDialog

Type $Username #1001
Type $Password #1002
Type "DB2" #1003
Click #1
```

Example 2**Windows Application Definition**

This example shows the use of the `-Raw` switch. This switch is not actually required in this instance, and is only there as an example.

```
# Calculator Is Active
Dialog
  Class #SciCalc
  Title "Calculator"
EndDialog
Type -Raw "15"
Type -Raw "+"
Type -Raw "20"
Type -Raw "="
```

Example 3**Windows Application Definition**

This example shows the use of the `-msg` switch. In this instance the switch is not actually required and is only shown as an example of the use of `Password` as the `-msg` argument.

```
# Calculator Is Active
Dialog
  Class #SciCalc
  Title "Calculator"
EndDialog
Type -msg $Password #480.
```

Example 4**Windows Application Definition**

The following syntax examples compare and contrast the use of the various `Type` command arguments.

```
type #1 "text"
```

Types `type text` into control with ID of 1

```
type #1 "text" -order
```

Types `text` into the first control found in the dialog when enumerating the children.

```
type #1 "text" -msg
```

Types `text` into the first control with an ID of 1 it finds within the set of windows allowing some time for the control to be created.

```
type #1 "text" -raw
type #1 "text" -focus
```

Ignores the unused parameter `#1`

Example 5**Windows Application Definition**

This example shows the use of the `-order` switch and demonstrates the possible “order” of the parameter.

```
type -order #1 "some text"  
type #2 "some text" -order  
type "some text" -order #3
```

Example 6**Web Application Definition**

This example uses the SecureLogin agent to automatically generate this application definition for the mail.yahoo.com Web site. This example shows the use of Password as the [`<Field Name>`] argument.

```
Type $Username  
Type $Password Password  
In the Application Definition above, the SecureLogin  
agent locates the first password field. The first Type  
command sends $Username to the field immediately before  
the password field. The second Type command sends  
$Password to the password field. The same Application  
Definition could be rewritten using absolute placement  
as shown below. In the following example, the Submit  
command is also used to automatically submit the page.  
Type $Username #1  
Type $Password #2  
Submit
```

5.2.88 Using the Type Command to Send Keyboard Commands

SecureLogin can send special keystrokes to Windows and Internet based applications to emulate the user's keyboard entry. The `Type` command can pass keystrokes through to the window that the application definition is using. These special commands include the ability to select menu items, send Alt key combinations, and send other keyboard combinations.

Special Key Commands

Type	Simulates
<code>\Alt+<key></code>	Pressing the Alt key plus the desired <code><key></code> .
<code>\Shift+<key></code>	Pressing the Shift key plus the desired <code><key></code> .
<code>\Ctrl+<key></code>	Pressing the Ctrl key plus the desired <code><key></code> .
<code>\LWin+<key></code>	Pressing the left Windows key plus the desired <code><key></code> .
<code>\RWin+<key></code>	Pressing the right Windows key plus the desired <code><key></code> .
<code>\Apps+<key></code>	Pressing the Application key plus the desired <code><key></code> .

Raw key commands

You can also use the `Type` command to send a combination of raw key commands. The [Section 7.2, “Windows Keyboard Functions,” on page 177](#) details the available keyboard sequences you can use with the `Type` command.

Type	Simulates
\ <xxx>	The format for sending a raw key command, where <xxx> represents the keyboard code.
\ 18+65	Pressing the Alt+A keys in sequence.

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 that implement this programming standard. The commands are the ones that you can use in the SecureLogin application definition 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 application definition language in the following format:

- ◆ TYPE @ command
- ◆ WAITFORTEXT "Log on:"
- ◆ Type \$username
- ◆ Type @T
- ◆ Type \$password
- ◆ Type @E

[Section 7.2, “Windows Keyboard Functions,” on page 177](#) details the available terminal emulator commands that you can use within a terminal emulator application definition.

5.2.89 WaitForFocus

Use With	Windows
SecureLogin Version	3.5 to 6.1 SP1
Type	Flow Control
Usage	WaitForFocus <#Ctrl-ID> [<Repeat-Loops>]
Arguments	<#Ctrl-ID> The ID number of the control with the focus. [<Repeat-Loops>] The number of repeat loops that runs.

Description	<p>Use the <code>WaitForFocus</code> command to suspend the running of the application definition until the <code><#Ctrl-ID></code> has received keyboard focus, or the <code><Repeat-Loops></code> expire. The <code><Repeat-Loops></code> is an optional value that defines the number of loop cycles to run. The <code><Repeat-Loops></code> value defaults to 3000 loops if nothing is set. After focus is received, the application definition continues.</p> <p>Set the figure to a negative number (for example <code>WaitForFocus "#1065" "-1"</code>) for the <code><Repeat-Loops></code> never to expire. If the <code><Repeat-Loops></code> is set to 0 (zero), it loops until the window defined in the <code>Dialog/ EndDialog</code> statement is given keyboard focus.</p> <hr/> <p>NOTE: Do not place <code>WaitForFocus</code> commands within <code>Dialog / EndDialog</code> statements.</p> <hr/>
Syntax Examples	<pre>WaitForFocus #301 WaitForFocus #301 "2000" WaitForFocus #301 "0" WaitForFocus #301 "-1"</pre>
Example 1	<p>Windows Application Definition</p> <p>This example has the <code>SecureLogin</code> waiting indefinitely for window <code>#301</code> to get focus. After the login dialog box is detected, it enters the user credentials.</p> <pre># Log on Dialog Box Dialog Class #32770 Title "Log on" EndDialog WaitForFocus #301 "-1" Type \$Username Type \T Type \$Password Type \N</pre>
Example 2	<p>This second example uses the <code>WaitForFocus</code> command to suspend the running of the application definition until <code><#Ctrl-ID> #15</code> is reached and a message box with "love" appears.</p> <pre>## BeginSection: "Global Script Configuration" ## EndSection: "Global Script Configuration" ## BeginSection: "Login Window" Dialog Class "Notepad" Title "Untitled - Notepad" EndDialog Setprompt "Optional:" # Here the correct id with the loops set to 0 waitforfocus #15 0 Set ?thu "love\me" RegSplit "(.*)\\(.*)" ?thu ?Domain ?User messagebox ?Domain ## EndSection: "Login Window"</pre> <hr/>

5.2.90 WaitForText

Use With	Terminal Launcher
SecureLogin Version	3.5 to 6.1 SP1
Type	Flow Control
Usage	WaitForText <Text>
Arguments	<Text> The text for which the application definition is waiting.
Description	<p>Use the <code>WaitForText</code> command to have the Terminal Launcher wait for the specified <text> to display before continuing. This command allows the user to wait for particular text to display before continuing. For example, waiting for a username field to display before attempting to type a username.</p> <p>The <Text> can appear anywhere on the terminal screen and is usually case sensitive (this depends on the terminal emulator itself). If the <Text> is written in the wrong case, the Terminal Launcher pauses and tries to find the correct <Text> in the correct case, until the terminal screen times out.</p> <p>If <code>WaitForText</code> is not working, try leaving the initial letter off the <Text> to avoid any conflict with case sensitivity. For example, <code>WaitForText login</code> works regardless of whether the word log on is presented on the terminal screen as Log on or log on. However, <code>WaitForText "Log on"</code> only works if the phrase log on is presented on the screen as "Log on".</p> <p>Also, some terminal emulators do not correctly match the text that is hard against the left margin of the window. Again, if you encounter this situation, try to match text without the leading character.</p>
Example	<p>Terminal Launcher Application Definition</p> <p>This command uses the <code>SecureLogin</code> to wait for the text "ogin" to appear on the emulator screen before entering the username. It then waits for "assword:" to display before entering the password.</p> <pre>WaitForText "ogin:" Type \$Username Type @E WaitForText "assword:" Type \$Password Type @E</pre>

Testing Application Definitions

6

This section contains the following information:

- ♦ [Section 6.1, “Using the SecureLogin Test Application,” on page 171](#)

6.1 Using the SecureLogin Test Application

To allow Administrators and other application definition writers to practice their application definition creation skills, the Password Test application is included in the software package. It is designed to replicate an application logon panel and supports the following processes:

- ♦ Initial log in
- ♦ Wrong password
- ♦ Password change

If you do not have the test application, contact Novell Technical Support.

The following example, application definition for the Password Test application, further explains the SecureLogin application definition principles.

6.1.1 Example Application Definition for the Test Application

The application definition for the PSL Password Test Application (`PasswordTest.exe`) provides an example of a typical Windows application definition, including error handling and changing the password. Remember, the password for this application is hard-coded to single when the application is closed and restarted. This can cause confusion when setting strong password policies and changing passwords. You must also create a password policy called `PwdTestPolicy`, according to the password policy defined in this application definition. The password policy must require a minimum of 6 characters, but no complex rules, in order to use single as a password.

Here is the sample application definition in its entirety. Following this application definition, [Section 6.1.2, “About the Application Definition,” on page 172](#) contains an explanation of what each section does.

```
# Set Password Policy
RestrictVariable $Password PwdTestPolicy
Application Definition continued on the next page
# ==== BeginSection: Log on ====
Dialog
Class "#32770"
Ctrl #1001
Title "Log on"
EndDialog
SetPrompt "Username =====>"
Type $Username #1001
SetPrompt "Password =====>"
Type $Password #1002
SetPrompt "Domain =====>"
Type $Domain #1003
Click #1
```

```

SetPrompt "Please enter your Username and Password to access NSL
Test. SecureLogin will remember and automatically log you on in
future. IT Helpdesk x4546"
# ==== EndSection: Log on ====
set it as a temporary variable, then
clear it
ReadText #65535 ?ErrorMessage
Click #2
# If log on failed, display the current stored Username and
Password and prompt the user to verify them, then retry log on
If "You have failed to log on." -In ?ErrorMessage
DisplayVariables "Log on to PSL Test Application failed. The
password for this app must be single when it first starts up. IT
Helpdesk x4563"
# Press Alt>F and L to invoke the Logon box so the User doesn't have
to.
Type -Raw "\Alt+F"
Type -Raw "L"
Type $Username
Type $Password
Type $Domain
EndIf
# ==== EndSection: Log on ====
# ==== Begin Section: Change Password ====
# Change Password Dialog Box
Dialog
Class "#32770"
Title "Change Password"
EndDialog
# Backup password, fill in the Old Username and Password, then
start the change password routine
Application Definition continued on the next page
Set ?PwdBackup $Password
Type $Username #1015
Type $Password #1004
ChangePassword ?NewPwd "Please enter a new password for the
application."
Type ?NewPwd #1005
Type ?NewPwd #1006
Click #1
# Change Password Successful message
Dialog
Class "#32770"
Ctrl #65535 "You have changed the password successfully."
Title "Change Successful"
EndDialog
# Clear Application owned message and accept new password
Click #2
Set $Password ?NewPwd
# ==== End Section: Change Password ====

```

6.1.2 About the Application Definition

You can use the same application definition to show what function each section performs. `Dialog/EndDialog` blocks define a Windows dialog box. When the dialog box appears, SecureLogin detects that this dialog box is based on the information found within the dialog block. The `Dialog/`

EndDialog block must contain enough information for the block to be unique, or the application definition runs when other dialog boxes owned by the same executable with the same information appear.

When SecureLogin detects that all the information between Dialog and EndDialog is contained in the dialog box on the screen (for example, the application login box, the change password box, or the failed logon box), it runs the application definition commands until it sees the next dialog statement or the end of the application definition, whichever is applicable. The order does not matter in Windows application definitions, because SecureLogin watches for all dialog boxes while the executable is running. Use a logical order for troubleshooting purposes.

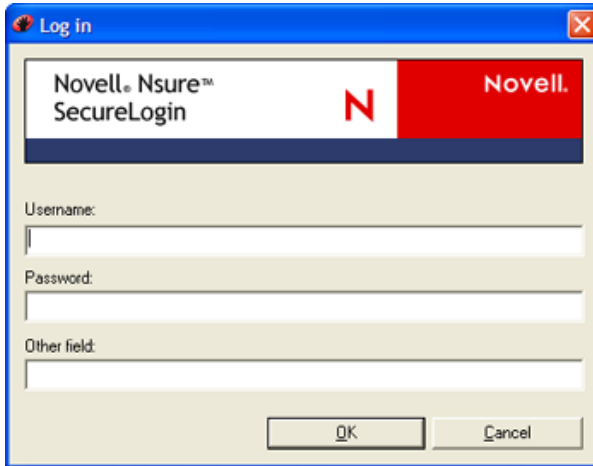
6.1.3 Dialog Boxes

The following application definition example shows screen captures of the relevant dialog boxes. You can use the Window Finder tool to gather information about the title of the window, class names, dialog IDs, and so on. Use the wizard to automate the application definition creation.

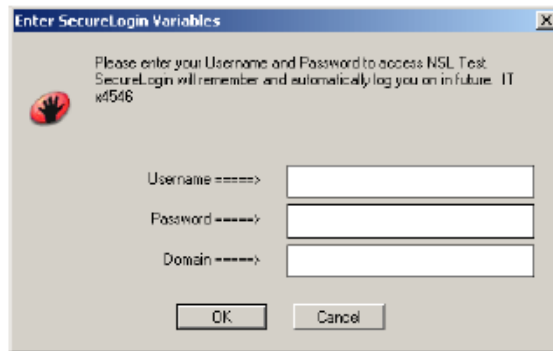
Table 6-1 Dialog Box Application Definition

Application Definition Section	Comments
# Set Password PolicyRestrictVariable \$Password PwdTestPolicy	This restricts the \$Password variable to comply with the Password Policy "PwdTestPolicy".
# ==== BeginSection: Log on ====Dialog Class "#32770" Ctrl #1001 Title "Log on"EndDialog	When PasswordTest.exe runs, SecureLogin watches for dialog boxes that appear and match the information defined between the Dialog/EndDialog commands. You can specify all values, or a few, as long as the information specified is unique to that dialog box.
SetPrompt "Username =====> "Type \$Username #1001 SetPrompt "Password =====> "Type \$Password #1002 SetPrompt "Domain =====>" Type \$Domain #1003 Click #1 SetPrompt "Please enter your Username and Password to access NSL Test. SecureLogin will remember and automatically log you on in future. IT Helpdesk x4546" # ==== EndSection: Log on =====	Type the stored (\$) Username variable into #1001, and so on. SetPrompt is used to customize the window the user sees when there are no credentials stored. When the user first runs an application that is newly enabled for single sign-on, SecureLogin prompts for their login credentials, and stores and remembers them for future login attempts.

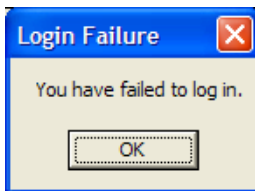
Application Definition Section**Comments**



The title is Log In.
The Class is #32770.
The *Username* field is Control ID #1001.
The *Password* field is Control ID #1002.
The *Other field* is Control ID #1003.
The OK button is Control ID #1.



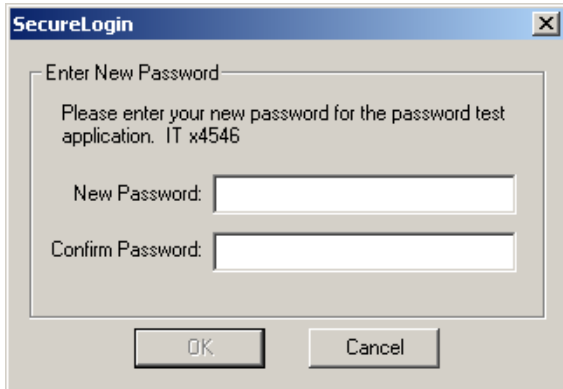
This dialog box is only displayed the first time the application definition is run by a user. It prompts the user to enter credentials for SecureLogin to store.
The `SetPrompt` command is used throughout the example application.



This is the login failure dialog box.
The title is Login Failure.
The class is #32770.
The OK button is Control ID #2.
The error message is Control ID #65535



This is the Change Password dialog box.
The *Username* field is Control ID #1015.
The *Old Password* field is Control ID #1004.
The *New Password* field is Control ID #1005.
The *Confirm New Password* field is Control ID #1006.
The OK button is Control ID #1.

Application Definition Section**Comments**

The `ChangePassword` command is used in the example application definition to display a dialog box for the user to enter a new password.

The dialog box is customized to provide more information for the user.

Reference Commands and Keys

7

- ♦ [Section 7.1, “Typing Keys,” on page 177](#)
- ♦ [Section 7.2, “Windows Keyboard Functions,” on page 177](#)
- ♦ [Section 7.3, “Terminal Emulator Commands,” on page 182](#)

7.1 Typing Keys

Do not type quotation marks before and after the keys. In this case the keys are taken literally, as shown in the following table.

Table 7-1 Example Typing Keys

For this Command	Type
Alt+Print Screen	\Alt+\ 44
Shift+Home	\Shift+\ 36
Shift+End	\Shift+\ 35

For more information about the `Type` command, see [Section 5.2.87, “Type,” on page 162](#).

7.2 Windows Keyboard Functions

The following tables list the Windows keyboard functions. You can use these functions in conjunction with the `Type` command by referencing the appropriate keyboard code.

Table 7-2 Windows Keyboard Functions

Function	Decimal	Comment
Left mouse button	1	
Right mouse button	2	
CTRL-Break	3	
Middle mouse button	4	
X1 mouse button	5	
X2 mouse button	6	
Backspace	8	
Tab	9	
Clear	12	5 on the keypad
Enter	13	

Function	Decimal	Comment
Shift	16	
Ctrl	17	
Alt	18	
Pause	19	
Cap Lock	20	
Escape	27	
Space	32	
PageUp	33	
PageDown	34	
End	35	
Home	36	
Left-arrow	37	
Up-arrow	38	
Right-arrow	39	
Down	40	
Select	41	
Execute	43	
Print	44	
Insert	45	
Delete	46	
Help Key	47	
0	48	
1	49	
2	50	
3	51	
4	52	
5	53	
6	54	
7	55	
8	56	
9	57	
A	65	

Function	Decimal	Comment
B	66	
C	67	
D	68	
E	69	
F	70	
G	71	
H	72	
I	73	
J	74	
K	75	
L	76	
M	77	
N	78	
O	79	
P	80	
Q	81	
R	82	
S	83	
T	84	
U	85	
V	86	
W	87	
X	88	
Y	89	
Z	90	
Left Windows Key	91	
Right Windows Key	92	
Application Key	93	
Sleep Key	94	
Keypad 0	96	
Keypad 1	97	
Keypad 2	98	

Function	Decimal	Comment
Keypad 3	99	
Keypad 4	100	
Keypad 5	101	
Keypad 6	102	
Keypad 7	103	
Keypad 8	104	
Keypad 9	105	
Keypad asterisk (*)	106	
Keypad plus sign (+)	107	
Keypad separator	108	
Keypad minus sign (-)	109	
Keypad period (.)	110	
Keypad slash mark (/)	111	
F1 key	112	
F2 key	113	
F3 key	114	
F4 key	115	
F5 key	116	
F6 key	117	
F7 key	118	
F8 key	119	
F9 key	120	
F10 key	121	
F11 key	122	
F12 key	123	
F13 key	124	
F14 key	125	
F15 key	126	
F16 key	127	
F17 key	128	
F18 key	129	
F19 key	130	

Function	Decimal	Comment
F20 key	131	
F21 key	132	
F22 key	133	
F23 key	134	
F24 key	135	
Num Lock key	144	
Scroll Lock	145	
Left Shift	160	
Right Shift	161	
Left Control	162	
Right Control	163	
Left Menu	164	
Right Menu	165	
Browser Back key	166	Applies to Windows 2000 +
Browser Forward key	167	Applies to Windows 2000 +
Browser Refresh key	168	Applies to Windows 2000 +
Browser Stop key	169	Applies to Windows 2000 +
Browser Search key	170	Applies to Windows 2000 +
Browser Favorites key	171	Applies to Windows 2000 +
Browser Start and Home key	172	Applies to Windows 2000 +
Volume Mute key	173	Applies to Windows 2000 +
Volume Down key	174	Applies to Windows 2000 +
Volume Up key	175	Applies to Windows 2000 +
CD Next Track key	176	Applies to Windows 2000 +
CD Previous Track key	177	Applies to Windows 2000 +
CD Stop Media key	178	Applies to Windows 2000 +
CD Play/Pause key	179	Applies to Windows 2000 +
Launch Mail key	180	Applies to Windows 2000 +
Media Select key	181	Applies to Windows 2000 +
Start Application 1 key	182	Applies to Windows 2000 +
Start Application 2 key	183	Applies to Windows 2000 +
;	186	Semi Colon/Colon

Function	Decimal	Comment
=	187	Equals/Plus Key
,	188	Comma/Less Than
-	189	Minus/Underscore
.	190	Period/Greater Than
/	191	Slash/Question Mark
`	192	Single Open Quote/Tilde
[219	Left Square/Curley Bracket
\	220	Back slash/Pipe
]	221	Right Square/Curley Bracket
'	222	Single Close Quote Double Quote
Play Key	250	
Zoom Key	251	

7.3 Terminal Emulator Commands

The following table lists the terminal commands in terminal emulator application definitions.

Table 7-3 Terminal Emulator Command Reference

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 Spaces)
@O	Space	@A@T	Print Presentation Space
@P	Print	@A@U	Cursor Up Fast

The Type Command	Meaning	The Type Command	Meaning
@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 Colours
@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
@l	PF21	@M@1	VT Numeric Pad 1
@m	PF22	@M@2	VT Numeric Pad 2
@n	PF23	@m@3	VT Numeric Pad 3

The Type Command	Meaning	The Type Command	Meaning
@o	PF24	@M@4	VT Numeric Pad 4
@q	End	@M@5	VT Numeric Pad 5
@s	SrcLk (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

Application Definition Commands for SNMP Alerts

8

Novell SecureLogin produces Simple Network Management Protocol (SNMP) for network monitoring software to trap. A simple application definition command is used to send the alerts.

You might need to copy the `LIBSNMP.DLL` file to the `Windows\System32` directory for SNMP support to work.

8.1 Creating an SNMP Alert

To produce an SNMP alert, place the following command in the application definition where you wish to create the alert:

```
Run C:\Progra~1\Novell\Secure~1\Slsnmp.exe <Community Name>
<Host IP Address> <Text>
```

- ◆ `<Community Name>` is the case-sensitive community name to which this computer sends trap messages.
- ◆ `<Host IP Address>` is the IP address of the SNMP host.
- ◆ `<Text>` is the text displayed as the message at the host.

Example Application Definition

```
Dialog
Class #32770
Title "Incorrect Password"
EndDialog
Run C:\Progra~1\ActivIdentity\Secure~1\Slsnmp.exe SNMPCommunity1
192.168.156.23 "PSL - Incorrect password in finance system."
MessageBox "You have entered an incorrect password. The
administrator has been notified. Please restart the application
and try again."
KillApp "PasswordText.exe"
```

