

Novell SecureLogin

2.5

www.novell.com

ADMINISTRATION GUIDE



N

Novell®

Legal Notices

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

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

This product may require export authorization from the U.S. Department of Commerce prior to exporting from the U.S. or Canada.

Copyright © 2001 Novell, Inc. All rights reserved. No part of this publication may be reproduced, photocopied, stored on a retrieval system, or transmitted without the express written consent of the publisher.

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

www.novell.com

Novell SecureLogin Administration Guide
August 2001
103-000131-001

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

Novell Trademarks

ConsoleOne is a trademark of Novell, Inc. in the United States and other countries.

eDirectory is a trademark of Novell, Inc.

GroupWise is a registered trademark of Novell, Inc.

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

NetWare is a registered trademark of Novell, Inc. in the United States and other countries.

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

Third-Party Trademarks

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

Contents

Preface	9
Conventions	9
1 Overview: Single Sign-on	11
The Security Problem	11
How Single Sign-On Fixes This Problem	11
The SecureLogin Solution	12
Requirements for an Effective Implementation.	13
SecureLogin Architecture	14
SecureLogin Components	17
Tools for eDirectory.	17
Administrator's Tools	17
The User Administration Tool.	18
Script Language	19
Corporate Login Scripts	19
Terminal Launcher	20
Single Sign-on for Windows Applications.	21
Internet Browsers.	21
Lotus Notes.	21
Mobile Single Sign-On	22
Background Authentication (Passticket)	22
2 Installing SecureLogin	23
Installing SecureLogin Components	23
Extending the eDirectory Schema	26
Granting Rights.	28
Automatically Granting Rights	28
Manually Granting Rights.	31
3 Using SecureLogin	33
Setting Up Caching and a Cache Passphrase	33
Capturing Passwords with the SecureLogin Wizard	34
Detecting a Windows Login Panel	34
Detecting a Web Login Page	35
Setting up Single Sign-on When Not Prompted	35
Managing SecureLogin.	37
Editing Login Details	37
Using the Login Wizard.	41
Setting Preferences.	43
Disabling Three Preferences	46

4	Administering Scripts	47
	Types of Scripts	47
	Administering Scripts for NT.	47
	Administering Scripts for eDirectory	50
	Using the SecureLogin Details Tab	50
	Using the SecureLogin Configuration Tab	51
	Using the SecureLogin Corporate Configuration Tab	53
	Finding Dialog Control IDs	54
	Predefined Applications	56
5	Setting Up Terminal Emulation	59
	Planning for Microsoft Terminal Server and Citrix MetaFrame	59
	Using SecureLogin Terminal Launcher	60
	Configuring an Emulator	60
	Configuring an Emulator Script	62
	Using Command Line Parameters.	64
	Configuring Backup Sessions	66
	Using Terminal Launcher With Non-HLLAPI Compliant Emulators	66
	Determining Which Session File To Use Automatically	68
6	Using Password Policies	71
	Creating a Password Policy	71
	Setting Advanced Criteria	73
	Using a Password Policy Script	74
7	SecureLogin Commands	77
	Commands	77
	ChangePassword	78
	Class	80
	Click	81
	Ctrl	82
	Delay.	82
	Dialog / EndDialog	83
	DisplayVariables	84
	EndScript	85
	If / Else /EndIf	86
	IfText / EndIfText.	88
	Increment / Decrement	89
	KillApp	90
	MessageBox	91
	Parent / EndParent	92
	PickListAdd	94
	PickListDisplay.	95
	ReadText	96

RegSplit	97
Repeat	98
RestrictVariable	99
Run	100
Strcat	101
Set	102
SetCursor	103
SetFocus	103
SetPlat	104
Setprompt	107
Title	108
Type	108
WaitForFocus	110
Waitfortext	111
A SecureLogin Script Language	113
Structuring and Executing Scripts	113
Using Variables	114
Substituting Variables	114
Using eDirectory Attributes	115
Using Variables	115
Using the Passticket Attribute	116
B Script and Keystroke Commands	117
Terminal Launcher HLLAPI Commands	117
Script Reference Guide	118
Windows and Internet Keystroke Commands	121

Preface

This document is for network administrators.

Novell[®] SecureLogin runs on any platform that is running NDS[®] eDirectory[™], including NetWare[®] 4.11 and later.

References to NDS eDirectory also imply references to NetWare 4.11 (which includes NDS) and later.

For the latest information and updates, visit the [Novell SecureLogin Web page \(http://www.novell.com/documentation/\)](http://www.novell.com/documentation/).

Conventions

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

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

1

Overview: Single Sign-on

The Security Problem

A major security problem that most organizations experience is the abundance of applications and systems that require individual user authentication. This is worsening as the number of Internet applications increases. As more companies switch to eBusiness solutions, their authentication requirements change. Businesses now have new requirements for high-end authentication.

Networking dependencies have also brought about new security requirements for authentication, with an increasing need to identify the end user. This requirement is becoming harder and harder.

Technology is advancing at a rapid pace and the difficulty of remembering usernames and passwords is becoming increasingly difficult. It is no longer reasonable for an organization to expect its users to remember long lists of username/password combinations, while still adhering to company security policies.

It is therefore not surprising that many users are reduced to writing username/password combinations on pieces of paper that are often left next to computers. This poses an extremely high security risk.

How Single Sign-On Fixes This Problem

Single sign-on fixes this problem by eliminating the need for users to remember their usernames and passwords beyond their initial network login. Single sign-on stores the usernames and passwords that each user needs and then automatically types them in for the user when required.

Single sign-on not only solves the security problem of users having to remember their login credentials, but also increases productivity because the

user no longer needs to type in a username and password. The computer does it for them. As a result, single sign-on greatly reduces the number of calls to help desks related to forgotten passwords.

The SecureLogin Solution

Novell® SecureLogin is comprised of multiple integrated security systems that provide authentication and single sign-on to networks and applications throughout an organization. The goal is to provide a single entry point to the corporate network and its resources for your users and at the same time increase security and improve corporate security policy compliance.

The separate single sign-on modules (components) of SecureLogin are designed for generic Windows*, Internet, and terminal emulator applications. SecureLogin's unique modular design allows it to be compatible with most new applications.

Security is an important feature of SecureLogin. It stores all user credentials encrypted in NDS® eDirectory™ and optionally caches details in an encrypted format on the local workstation. Only the user whom the details are stored for is able to unlock the encrypted data. For example, a network administrator with full rights is not able to see what a user's password is for Internet banking.

SecureLogin is extremely easy to use. With the use of wizards, corporate scripts, and predefined applications, administrators are able to intuitively configure SecureLogin for use in the corporate network from a central point using eDirectory. SecureLogin also includes a workstation administration tool that allows users to view their single sign-on details and, if permitted by the administrator, add new applications and Web sites for single sign-on.

SecureLogin employs two methods of fault tolerance. One method uses local encrypted caching to ensure that network downtime does not effect single sign-on performance. Even if the corporate network is down, by the use of caching, application logins continue uninterrupted. A second method allows for scripting to cater to different login conditions and errors during login.

Local encrypted caching also allows SecureLogin to maintain single sign-on integrity for all mobile and remote users, regardless of network connectivity. If permitted by the administrator, mobile users can update their single sign-on credentials when disconnected from the network and update eDirectory with these details when they are next attached.

Because SecureLogin is an eDirectory-enabled product, users can roam wherever eDirectory is. They can

- ◆ Log in from anywhere and get the same capabilities as if they were at their own desks.
- ◆ Log in and log off quickly, because they only authenticate to eDirectory, not to Windows itself.
- ◆ Roam the enterprise, logging into several different machines during the day.
- ◆ Work on a notebook or laptop in a disconnected mode, because their login credentials are saved to a local, encrypted cache.
- ◆ Securely use a shared, kiosk-type workstation, where many people log in temporarily for quick work and then log out.

Single sign-on has two ultimate goals:

- ◆ Automate the repetitive manual login processes so that logging in is seamless to the user
- ◆ Lower overall maintenance costs for network administrators

Requirements for an Effective Implementation

A successful single sign-on system must meet the following seven requirements. SecureLogin was designed with these requirements in mind.

- ◆ Meet the changing needs of large organizations.

The implementation must allow new software to be easily installed and configured for single sign-on.

- ◆ Easily accommodate mobile users.

Remote and roaming users must be able to access their single sign-on credentials and update them if necessary.

- ◆ Provide ease of management, rapid deployment, and fault tolerance.

The single sign-on system must run efficiently and be easy for users to operate. It must also be easy for administrators to control and maintain.

- ◆ Employ industry standards and open architecture.

The single sign-on system must be compatible with most existing software.

- ◆ Be secure.

The password storage and playback mechanism must not allow for stealing secrets. The usernames and passwords must be encrypted and stored in a secure database.

- ◆ Be cost effective.

The single sign-on system must save money and reduce the cost of ownership.

- ◆ Be seamless to the user.

The second time a user logs in to an application, the application should look the same as the first time. Subsequent attempts to open the application should involve no user interaction for authentication.

SecureLogin Architecture

SecureLogin is an authentication and single sign-on suite of applications. The application suite consists of an eDirectory database, authentication server, NetWare[®] and Windows NT* client authentication plug-ins, and single sign-on modules for most popular applications.

SecureLogin works by keeping a record of user authentication credentials and instructions about how to use these credentials. SecureLogin stores these records in eDirectory. At run time, SecureLogin queries these records and then supplies the correct authentication details for the user.

The SecureLogin script language is a key feature of SecureLogin single sign-on, enabling the product to be compatible with almost all network environments and applications. The script language has the following advantages:

- ◆ Allows an administrator to define single sign-on methods for almost any Windows, mainframe, Internet, intranet, Terminal Server or UNIX* application.
- ◆ 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.

All SecureLogin data is stored in one or more database objects residing in eDirectory. SecureLogin's first task on startup is to locate these objects in the eDirectory and cache their contents in memory for later use by the workstation SecureLogin single sign-on agent.

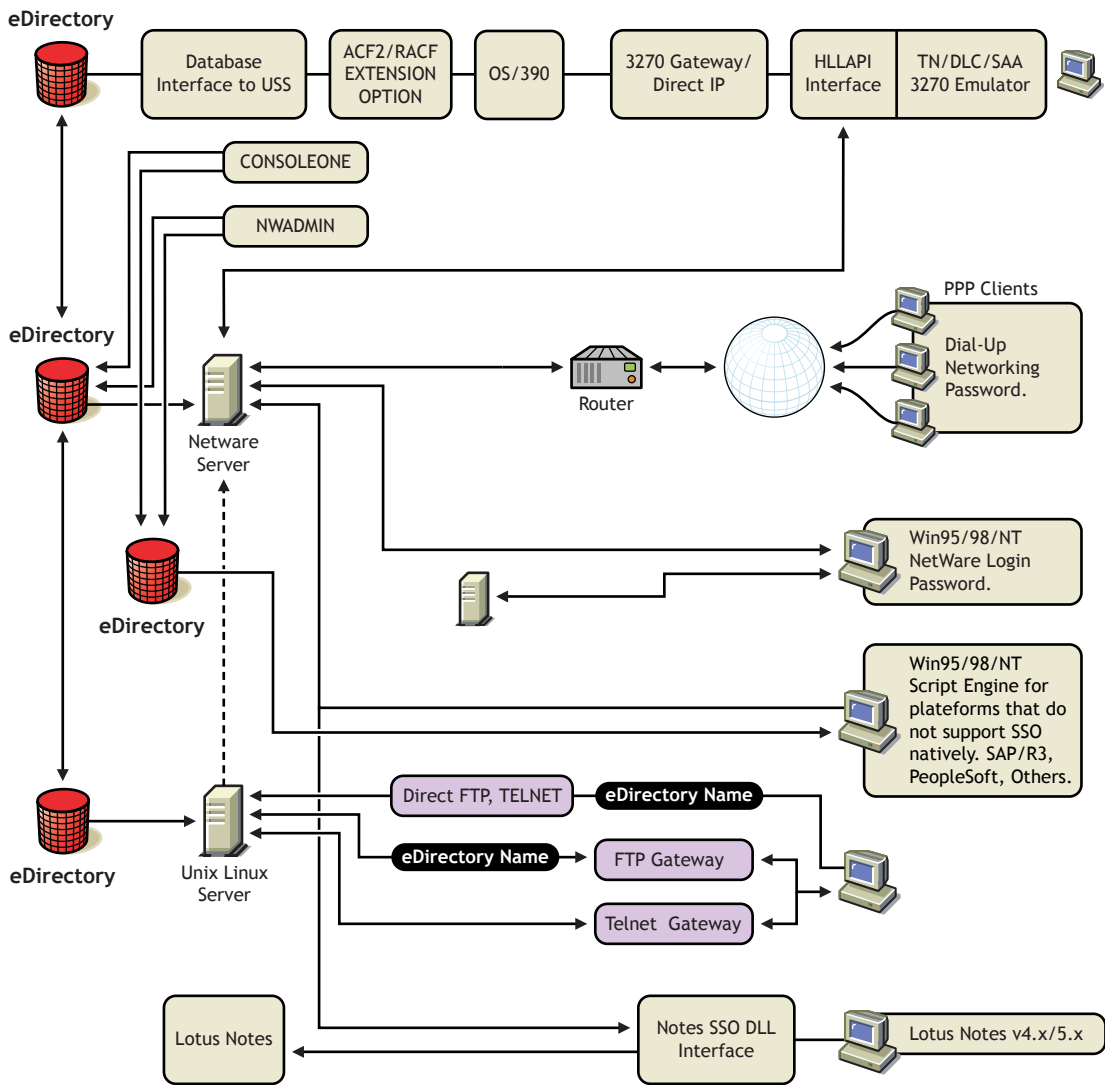
SecureLogin allows you to define which applications are to be enabled for single sign-on. This option gives you

- ◆ Full control of which applications can be used for single sign-on
- ◆ The ability to update the entire eDirectory database with a new application login script by updating a single object.

These corporate scripts are stored in a Container object rather than individual User objects. For users, the result is a less complex system. For you as the administrator, the improved login mechanisms provide

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

The following figure illustrates SecureLogin architecture.



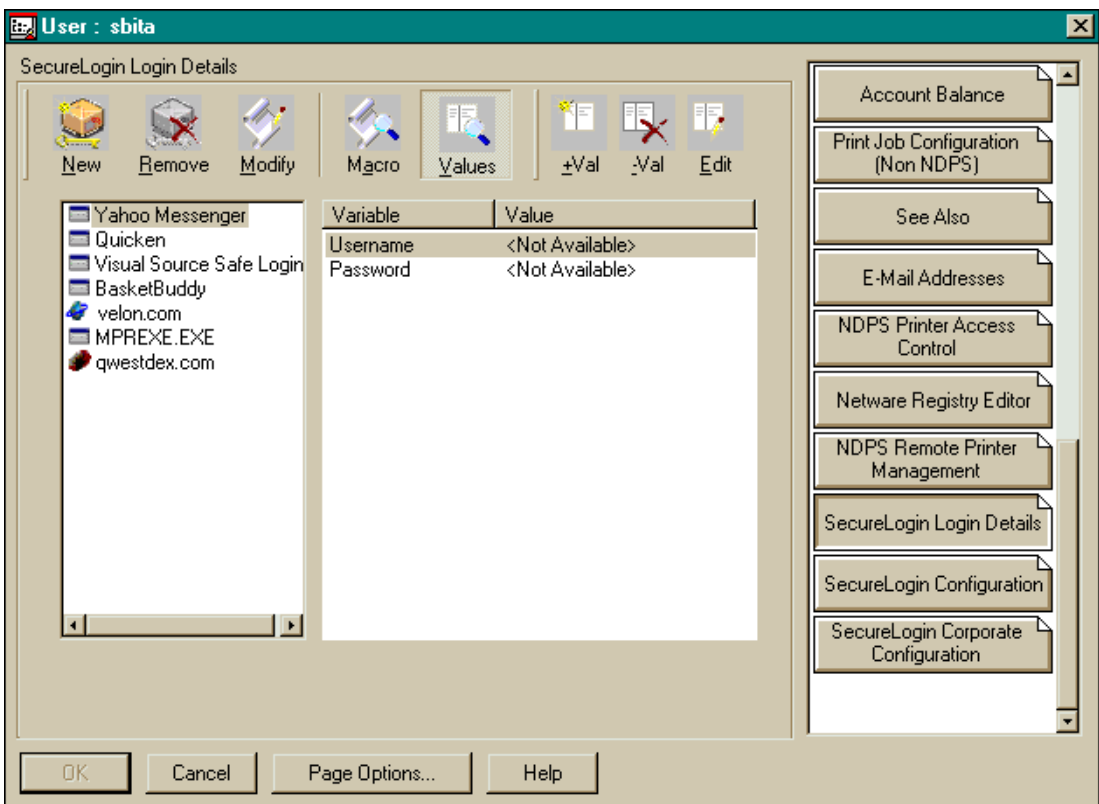
SecureLogin Components

Tools for eDirectory

SecureLogin leverages your existing eDirectory to provide user, organization, and application administration for your single sign-on solution. With the SecureLogin administration tools, you can centrally manage users and corporate single sign-on applications and configurations.

Administrator's Tools

You are able to set SecureLogin preferences for the users at either the User object level or at the container level. To manage these eDirectory objects, you can use either NetWare Administrator or ConsoleOne™. The following figure illustrates the SecureLogin Login Details, SecureLogin Configuration, and SecureLogin Corporate Configuration tabs.



The User Administration Tool

You can also administer SecureLogin through the user administration tool.



The workstation user administration tool is used to

- ◆ View existing applications and Web sites that single sign-on is enabled for.
- ◆ Modify passwords to existing single sign-on enabled sites.
- ◆ Add new applications for single sign-on
- ◆ Control preferences on how the product will behave

Installed with SecureLogin, this tool enables users to view their single sign-on credentials. Also, you can permit users to administer their own credentials.

You can access this tool by clicking the SecureLogin icon on the workstation's task bar. The following figure illustrates this icon:



Using eDirectory, you can enable or disable all of these functions for individual users and the entire organization.

Script Language

SecureLogin uses a script language to provide a very 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 a script to enter the username, password, and background authentication information.

The script language uses individual application scripts to retrieve and enter the correct login details. These scripts are stored and secured within eDirectory to ensure maximum security, support for single point administration, and manageability.

The script language can be used to automate many login processes, such as multi-page logins and login panels requiring other information that can be stored in eDirectory (such as surname and telephone number). The script language also contains the commands required to

- ◆ Automate password changes on behalf of users
- ◆ Request user input when it is required

Corporate Login Scripts

SecureLogin is designed for large networks. It supports the ability to use eDirectory to centralize the setup of the single sign-on applications. This feature is referred to as Corporate Login Scripts.

A corporate login script can be stored in either a file system or in a Container object located in eDirectory. This feature gives you the ability to write and define single sign-on scripts once for the whole organization, while still allowing for customized subordinate containers and User objects. This customization significantly reduces the effort and complexity of enterprise deployment.

If a subordinate object has a different script for the same application defined locally, the local copy will be used instead of the version that is on the higher object. If a script is defined on a User object with the same name as a script defined on a Container object, or if there are two scripts with the same name on different level Container objects, the script from the subordinate object will always be used instead of the script in the higher level object. This strategy allows for specialization in corporate scripts.

For more detail on scripting, see [Chapter 4, “Administering Scripts,” on page 47](#) and [“Script Reference Guide” on page 118](#).

Terminal Launcher

Terminal Launcher enables you to easily launch terminal emulation sessions and to run a script within those sessions.

The script is stored within eDirectory, which makes it more secure than less generic scripts that are written in a particular language for a particular emulator. These scripts are designed to be compatible with many different emulators.

With the use of corporate scripts, the Terminal Emulator Launcher is very powerful. It can be used to provide shortcut icons to mainframe or UNIX applications, removing the need for user intervention.

You access Terminal Launcher from Start > Programs > Novell SecureLogin > Terminal Launcher.

Mainframe Types

OS/390: ACF2 and RACF

The OS/390 component allows clients to authenticate to a mainframe using a fixed password. It also provides background authentication if the user has logged onto the LAN and the LAN has been set up as a trusted partner.

The SecureLogin ACF2 and RACF interfaces (known as SAF exits) are called from ACF or RACF to validate a user's account and password. The Exit reads the user's authentication information from InfoStorage (the mainframe database), validates the user-provided details, and returns an ACCEPT or DENY to ACF2 or RACF.

The InfoStorage database is a copy of the user's authentication data held inside eDirectory. The SecureLogin ACF2/RACF is composed of two components:

- ♦ A UNIX System Services application, which synchronizes eDirectory data with the mainframe security database
- ♦ The ACF2/RACF validation exit

AS/400

SecureLogin supports the AS/400 with its Universal Single Sign-On client and can use the script engine to automate complex multi-stage logins as required.

Single Sign-on for Windows Applications

The component for Windows applications enables most Windows applications to use single sign-on. This component works with

- ♦ All Windows applications that have generic username and password prompts
- ♦ Most specialized applications that have very complex logins

Internet Browsers

The Microsoft* Internet Explorer and Netscape* components enable applications that are accessed through these browsers to use single sign-on.

This component also enables sites using http dialogue authentication to use single sign-on.

Lotus Notes

The SecureLogin Lotus Notes* component enables you to use single sign-on with Lotus Notes. This component is a more specialized version of the Windows applications single sign-on component and is designed so that you do not even notice you have switched over to single sign-on (apart from the lack of login screens). A Lotus Notes SecureLogin DLL, which tightly integrates with the Lotus Notes authentication system, is installed on each workstation.

After installation, the next time you authenticate to Lotus Notes you actually type your password into a SecureLogin panel designed to look like the Lotus Notes password box. This will be the last time you have to enter your password into Notes.

The next time you are required to authenticate, SecureLogin communicates with Lotus Notes in the background. The password box to log in never appears. At the end of the password expiration period, SecureLogin can prompt for a new password or automatically populate the password field.

SecureLogin supports password expiration in Notes and, as with all applications, can be set up to automatically generate a random password, based on a password policy. In addition to controlling single sign-on, this component supports

- ♦ Multiple ID files for each user
- ♦ The ability to exclude certain administrative IDs from being enabled for single sign-on

Mobile Single Sign-On

Taking advantage of eDirectory architecture, SecureLogin allows users to roam with their authentication details. Because there are no workstation dependencies, users can move freely from office to office. Their credentials follow them.

By using the local encrypted cache, SecureLogin also allows notebook users access to single sign-on.

Background Authentication (Passticket)

SecureLogin includes a method of signing on to back-end systems. This method does not involve the use of a password. Instead, it uses a cryptographic key process to securely authenticate the user to the remote system. This technology is referred to as Passticket single sign-on or background authentication.

Background authentication means that rather than simply typing in the username and password for a user, SecureLogin can effectively take over the authentication process of the application by using a shared cryptographic key between different platforms (such as the LAN and mainframe).

This method can only be achieved on applications that give programmers interfaces into their products, and the interfaces allow Passticket technology.

SecureLogin currently supports a number of platforms with background authentication, including UNIX, mainframe, and Internet servers such as Netscape Enterprise Server*.

2

Installing SecureLogin

This section helps you complete the following tasks:

- ♦ Install SecureLogin components
- ♦ Extend the NDS[®] eDirectory[™] schema

Installing SecureLogin Components

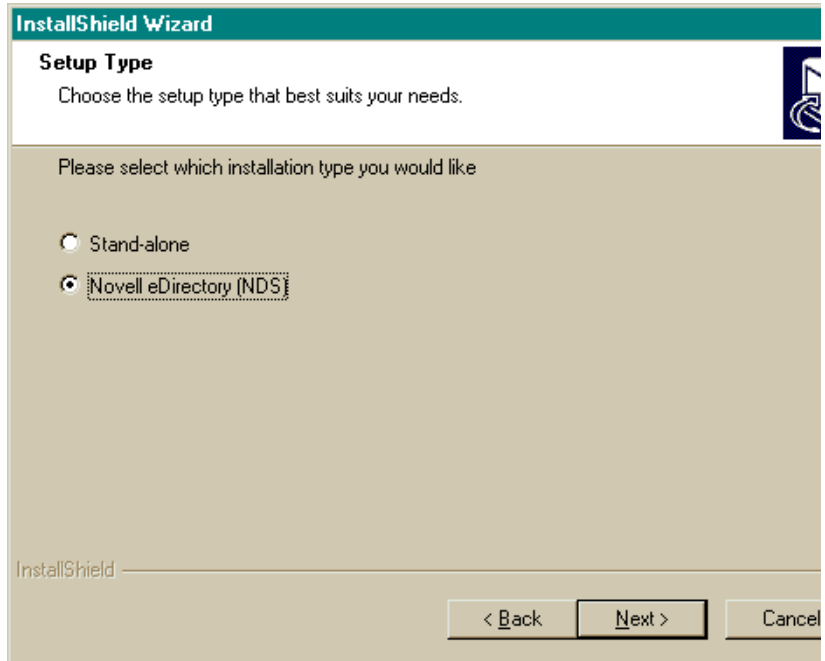
Novell[®] SecureLogin thrives with NDS eDirectory running on Windows* NT*, NetWare[®] 4.11 or later, Solaris*, and Linux*.

You can install SecureLogin on a Windows 95, Windows 98, Windows ME, Windows NT 4.0, or Windows 2000 workstation.

Windows 95, Windows 98, and Windows ME workstations should have Novell Client[™] 3.21 or later.

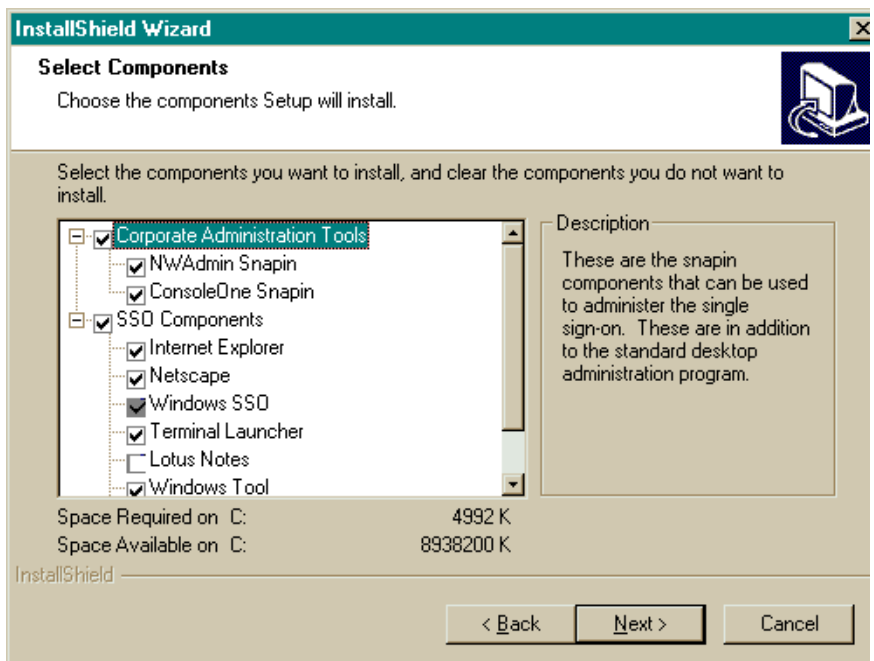
Windows NT and Windows 2000 workstations should have Novell Client 4.71 or later.

- 1** From the SecureLogin CD's INSTALL directory, run SETUP.EXE
- 2** At the InstallShield Wizard's Welcome screen, click Next.
- 3** At the Setup Type screen, select the Novell eDirectory (NDS) option > click Next.



4 Select a destination folder > click Next.

5 Select components to install > click Next.

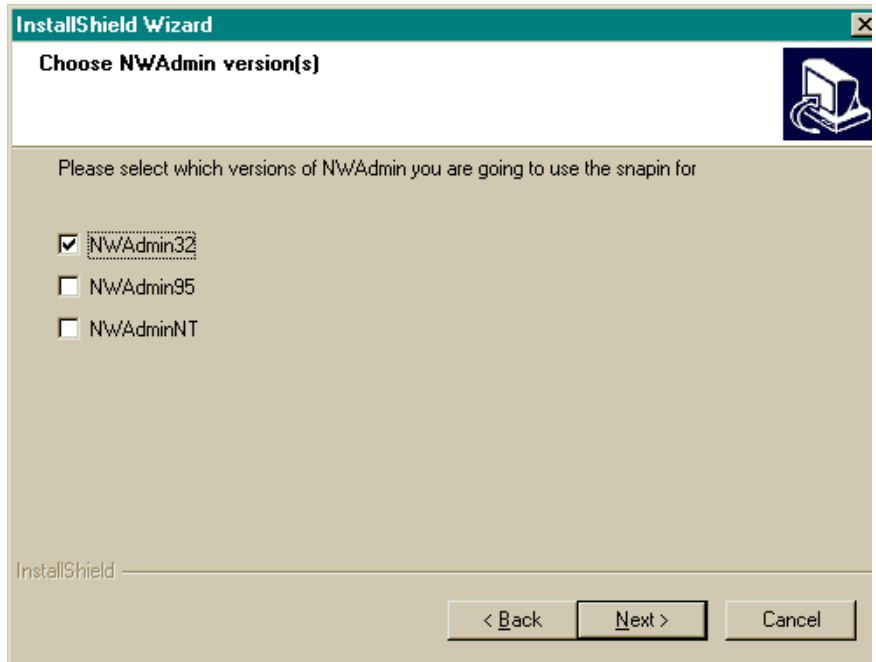


You can administer SecureLogin from NetWare Administrator or ConsoleOne™.

You must install the Windows SSO component. Install other components according to software used on your network.

If you select Lotus Notes* and the location of the NOTES.INI file is something other than your Windows directory, InstallShield will prompt you for the file's location.

- 6** Follow the on-screen prompts to install selected components and icons.
 - 6a** (Conditional) If you selected NWAdmin Snapin, select a version of NetWare Administrator that matches the platform running on the workstation.



Platform	Version
Novell Client 32	NWAdmin32
Windows 95	NWAdmin95
Windows NT	NWAdminNT

- 6b** (Conditional) If you selected ConsoleOne Snapin, select a target directory where the files snap-in files will be installed < click Next.
 - 6c** (Conditional) If this is the first installation of SecureLogin, check the Extend Directory Schema check box > click Next.
- See “[Extending the eDirectory Schema](#)” on page 26.

Extending the eDirectory Schema

For SecureLogin to be able to save user single sign-on information, the eDirectory schema must be extended. Therefore, on the first workstation that you install into an NDS eDirectory tree, select Extend Directory Schema.

Do not select Extend Directory Schema for subsequent installations. You only extend the eDirectory tree schema once for SecureLogin.

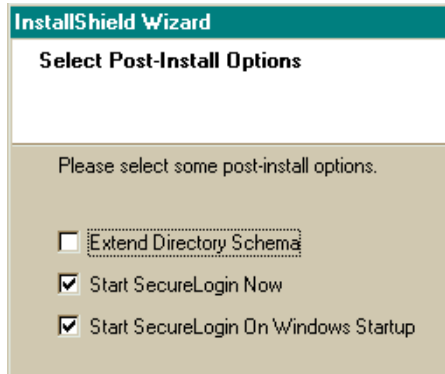
To extend the schema of a given tree, you must have sufficient rights over the [root] of the tree.

If required, the workstation installation program will automate the extension of eDirectory.

Before attempting to update the schema, the InstallShield program checks to see whether the schema has been extended. To prevent InstallShield from repeatedly looking to update the schema, leave the Extend Directory Schema check box unchecked.

This option grants existing users rights to the SecureLogin attributes on the User object. Therefore, unless you have modified the user template object to give new users these rights (or otherwise made similar arrangements), you might need to run this tool whenever you add new SecureLogin users.

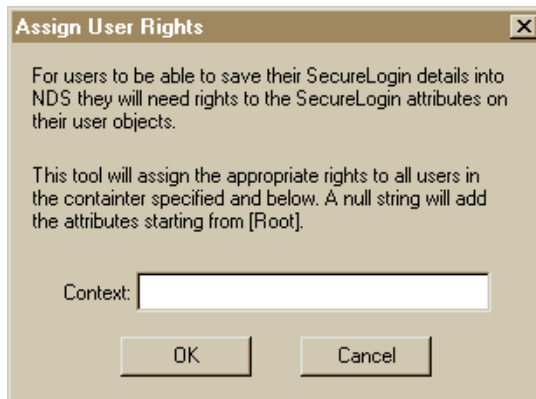
- 1** At the Select Post-Install Options screen, select Extend Directory Schema > click Next.



The extension may take some time to filter throughout your network, depending on the size of your network and the speed of the links.

- 2** Enter an eDirectory context so that SecureLogin can assign rights to User objects.

If this is the first installation of SecureLogin, you will be prompted to define a context where you would like the User objects' rights to be updated, allowing users access to their own single sign-on credentials.



If the installation program displays a message similar to FFFFD9D, you have probably entered an incorrect context or included a leading dot in the context.

- 3** Select a program folder for SecureLogin icons > click Next > Finish.

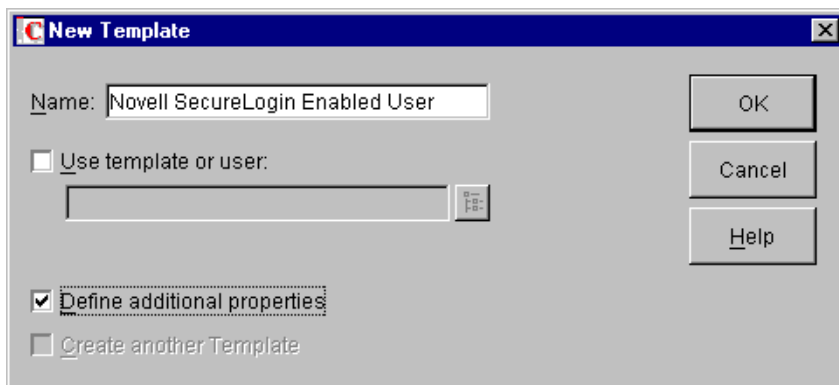
Granting Rights

You can automatically or manually grant rights to User objects created after Novell SecureLogin is installed.

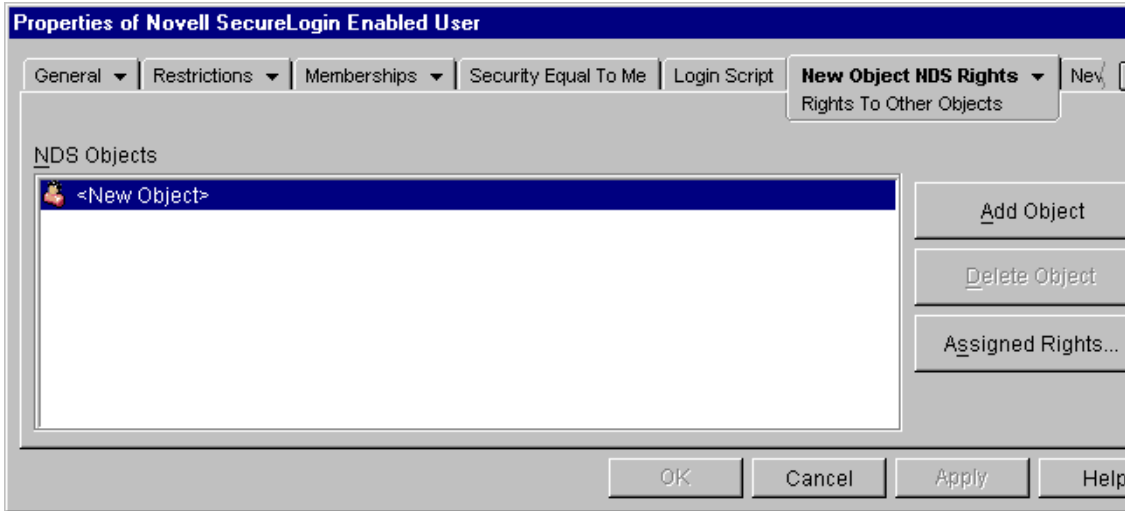
Automatically Granting Rights

You can create a user template that automatically grants rights to required attributes. The following steps refer to ConsoleOne.

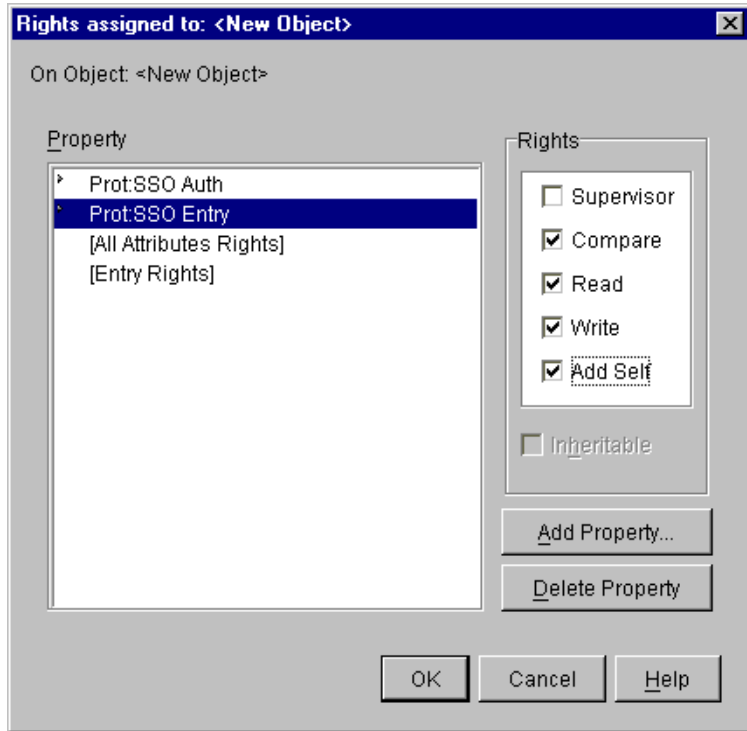
- 1** Select an O or OU Container object that will contain the Template object.
- 2** Select to create a new object of the type Template.
- 3** Name the template > check the Define Additional Properties check box > click OK.



- 4** At the properties box, navigate to New Object NDS Rights > Rights To Other Objects.
- 5** Select New Object > click Assigned Rights.



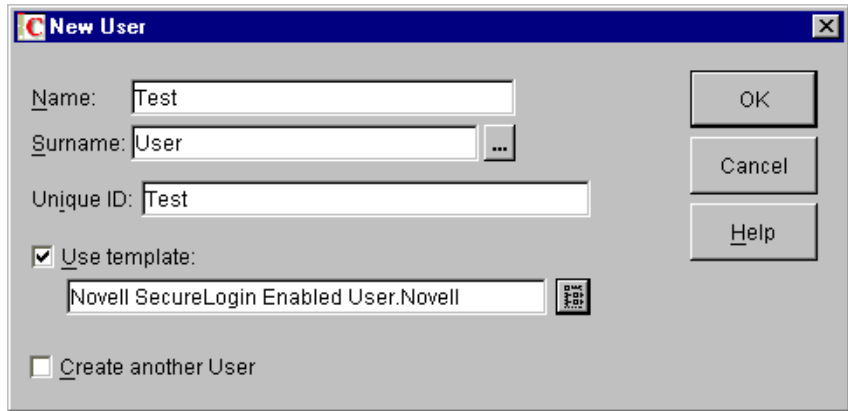
- 6** Click Add Property > select the attribute Prot: SSO Auth > click OK.
- 7** Check the Compare, Read, Write, and Add Self check boxes.
- 8** Repeat Steps 6 and 7 for the attribute Prot: SSO Entry.
NOTE: Do not add the attribute Prot: SSO Profile.



9 Exit by clicking OK > OK.

To use the new template:

- 1** Create a new User object.
- 2** Check the Use Template check box > click the Browse button.



- 3 Select the Template object that you created > click OK.

Manually Granting Rights

You can manually grant rights to users created after Novell SecureLogin has been installed. Run SCHEMA.EXE, which is typically located in the C:\PROGRAM FILES\NOVELL\SECURELOGIN directory.

This program extends the schema and grants rights to existing users listed in the installation.

3

Using SecureLogin

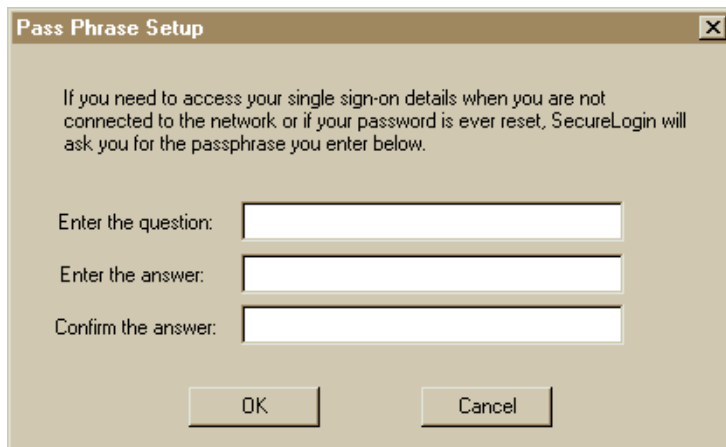
You use SecureLogin to capture login information and save that information to your workstation and eDirectory™.

Using a SecureLogin administrative tool, you can view and edit your saved information.

Setting Up Caching and a Cache Passphrase

To allow single sign-on to continue to function seamlessly even when the network has problems, SecureLogin can store on the workstation an encrypted cache of your single sign-on details. This cache includes all corporate scripts as well as information defined on your User object.

The cache file is encrypted when you create a user-defined passphrase and password. The following figure illustrates the dialog box that creates a passphrase and password:



The image shows a dialog box titled "Pass Phrase Setup" with a close button (X) in the top right corner. The dialog contains the following text and input fields:

If you need to access your single sign-on details when you are not connected to the network or if your password is ever reset, SecureLogin will ask you for the passphrase you enter below.

Enter the question:

Enter the answer:

Confirm the answer:

At the bottom of the dialog are two buttons: "OK" and "Cancel".

When the cache file is created for the first time, SecureLogin prompts you to provide this password. (Make sure that the password is easy to remember.) This password is only required if

- ◆ The network connection is lost
- ◆ You are using a notebook computer and are out of contact with the network
- ◆ The network administrator resets your eDirectory password

The cache files are stored in the PROGRAM FILES\NOVELL\CACHE directory and are triple Data Encryption Standard (3DES) encrypted.

If you forget the cache password, you will have to delete and recreate the cache files. SecureLogin automatically recreates cache files, provided you are authenticated to the network. You will be prompted for a new passphrase and password when SecureLogin recreates the cache files.

You can turn off caching functionality. In NetWare® Administrator, select the SecureLogin Configuration tab > uncheck the Use Cache Files check box.

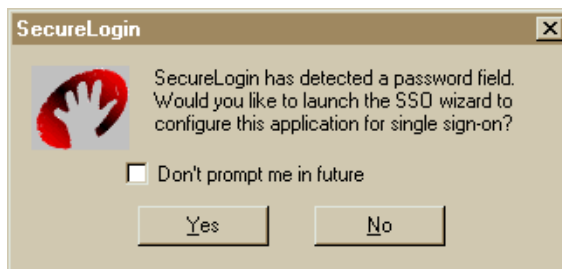
Capturing Passwords with the SecureLogin Wizard

SecureLogin provides wizards that help you generate login scripts and store variables. These scripts avoid lengthy setup periods.

Scripts store the login name, password, and any other information fields required for authentication. Scripts are stored in the local database and in eDirectory.

Detecting a Windows Login Panel

After detecting a Windows login panel, SecureLogin displays a dialog box so that you can enable single sign-on to this product.



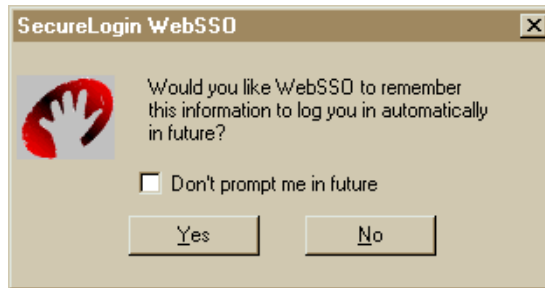
You can choose not to enable single sign-on for this application as well as configure SecureLogin so that this setup window does not display again for this application. If you select not to be prompted, SecureLogin generates a blank script for this application and doesn't prompt you again. If you decide to set up single sign-on, SecureLogin launches a wizard that guides you through the setup.

For some applications, SecureLogin may contain a pre-built script. In this situation, SecureLogin prompts you for your login credentials and enables single sign-on to that application. For more information, see [“Predefined Applications” on page 56](#).

The next time you open the application, SecureLogin authenticates for you.

Detecting a Web Login Page

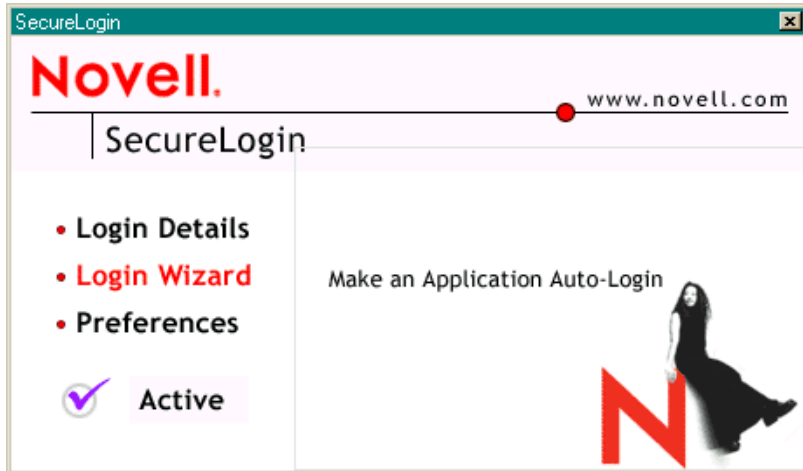
When you navigate to a Web page that has a login field and then submit your login information, SecureLogin prompts you whether to enable single sign-on for the site.



If you select Yes, SecureLogin extracts the login information from the Web page and stores it for future single sign-on authentication.

Setting up Single Sign-on When Not Prompted

If you earlier selected “Don't prompt me in future” or for some reason do not want to launch the application to set up single sign-on, you can launch a wizard manually by clicking Login Wizard on the user management tool.



- 1 Right-click the SecureLogin icon on the system tray.
- 2 Select Login Wizard.
- 3 Enter a description for the application (platform) > select the type of application > follow the on-screen prompts.



Managing SecureLogin

Each workstation running SecureLogin has an administration tool.

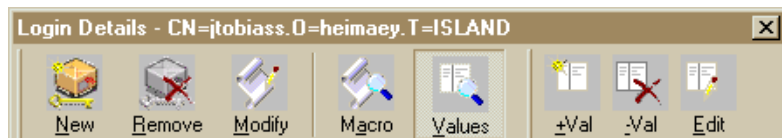


To launch this tool, double-click the SecureLogin icon on the system tray.

You can use this tool to manage users' single sign-on credentials and configure SecureLogin preferences. This tool can only alter the current user's SecureLogin information. To create corporate scripts you must use the NetWare Administrator or ConsoleOne™ snap-ins.

Editing Login Details

The Login Details option enables you to view, edit, add, or delete login information. The following figure illustrates buttons that enable you to complete these tasks.



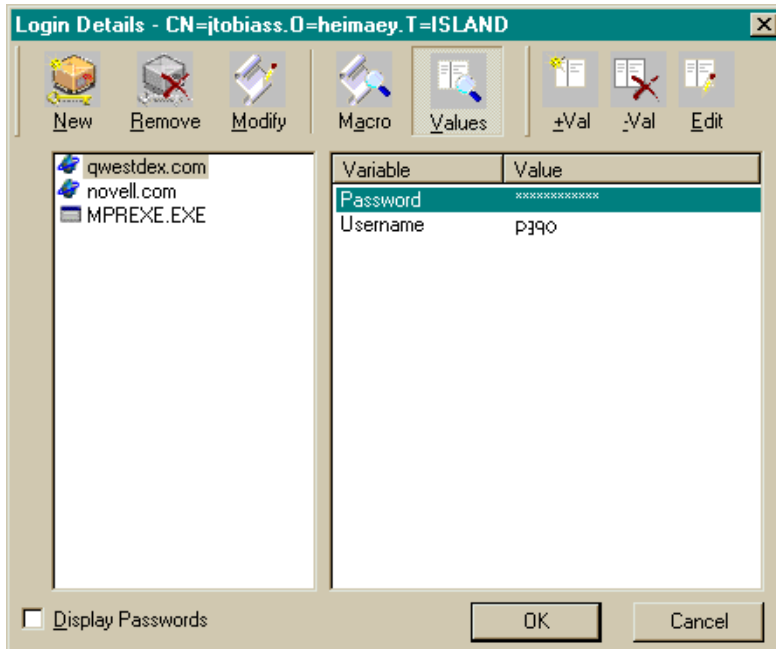
To access the Login Details screen, select the Login Details option on the administration tool, or right-click the SecureLogin icon on the system tray > select Login Details.

If you make a mistake while adding, editing, or removing details, click Cancel to close the application without saving any changes.

Editing a Platform

The platform column is the leftmost list on the Login Details screen. It contains a list of different platforms or applications that SecureLogin is currently authenticating to.

In the following illustration, three platforms are enabled:



Each listed platform name has an icon next to it. These icons identify the platform type:

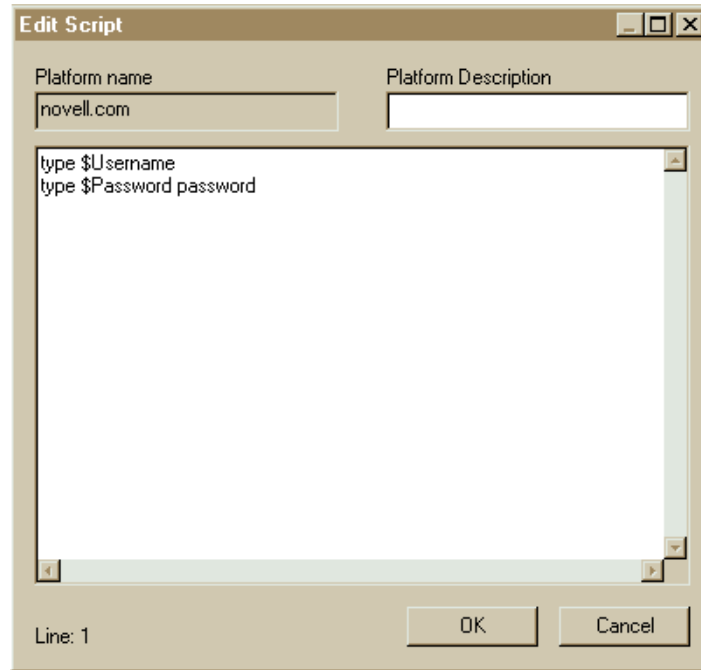
- ◆ Web Application
- ◆ Windows Application
- ◆ Terminal Emulator Application
- ◆ Default SecureLogin type

Some additional modules (for example, Lotus Notes) might allow for their own specific type.

When you select a platform, information in the Variable and Value columns is updated.

To edit the script for a platform, double-click the platform or click Modify.

The following figure illustrates a script.



Adding a Platform

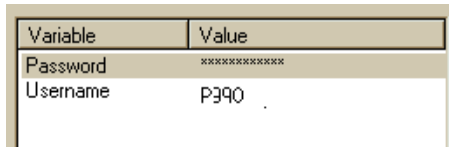
The New button enables you to add a platform or application to the SecureLogin list. SecureLogin then automatically authenticates and logs you in. This option is for administrators and advanced users.

This procedure is not necessary for most applications because it is easier to use the wizards to set up single sign-on for a platform.

- 1** Click New.
- 2** In the Platform Name box, provide a name for the application.
For Windows Application script types, you must use the application executable name as the platform name. For Internet Page script types, you must specify the URL address as the platform name.
- 3** Select a Script Type > click OK.
- 4** Enter a script.

Editing a Variable or Value

As the following figure illustrates, the Variable column usually just contains the password and username for a particular application.



Variable	Value
Password	*****
Username	p390

However, in some more complicated applications, there may be other variables too.

This example has two variables, Password and Username. The script for this platform has the following line:

```
type $Username
```

The value for this variable is p390. At run time, the value p390 will be substituted with \$Username. This substitution allows corporate scripts to work. A corporate script might include *type \$Username*. Depending on who is running the script, an individual username would be substituted.

To edit a value for a variable, double-click it, or select it > click Edit.

To delete a variable and accompanying value, select the variable > click -Val.

For more information on variable substitution, see [Appendix A, “SecureLogin Script Language,” on page 113](#).

Adding and Removing Variables

You can add a new variable to the list of variables of the currently highlighted platform.

- 1 Click +Val.
- 2 Enter a variable name
- 3 Enter a value > click OK.

To remove a variable and accompanying value, select the variable > click -Val.

Releasing an Application from SecureLogin

To delete an application from the platform list, select it > click Remove.

Viewing SecureLogin Macros

To view SecureLogin macros, click the Macro button, which does the following:

- ◆ Toggles the view from the Variable view to the Script view
- ◆ Disables the variable buttons (+Val, -Val, Edit)
- ◆ Replaces the variable and value columns with a read-only edit box containing the current script

To toggle back to the Variable view, click Values.

Displaying Passwords

By default, the Login Details screen displays passwords as xxxxxxxx. You can view the actual password.

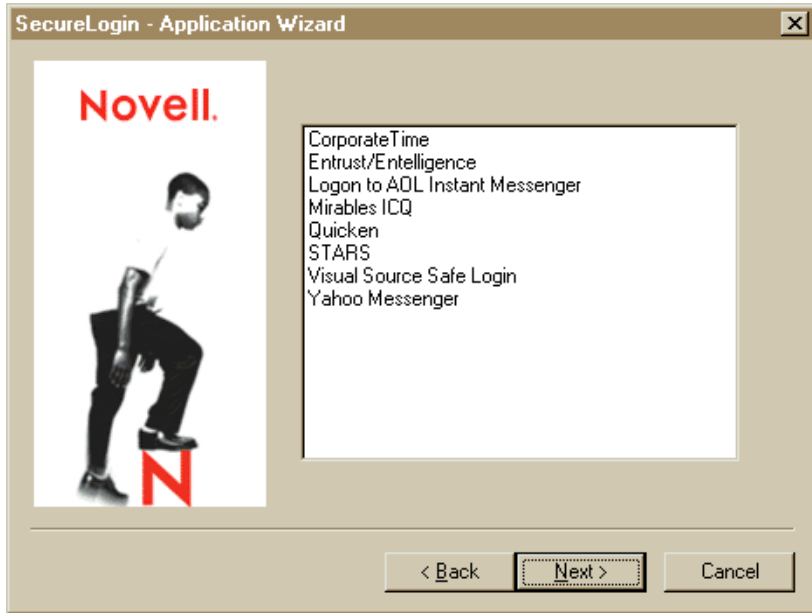
- 1** Select the application.
- 2** Check the Display Passwords check box.

Using the Login Wizard

The Login Wizard generates scripts for Windows and Web applications. You can also use this wizard to run predefined scripts.

Using Predefined Scripts

- 1** At the Novell® SecureLogin tool's main screen, select Login Wizard.
- 2** Enter a description, for example, Quicken.
- 3** Select Predefined Script > click Next.
- 4** Select from the list of predefined applications.



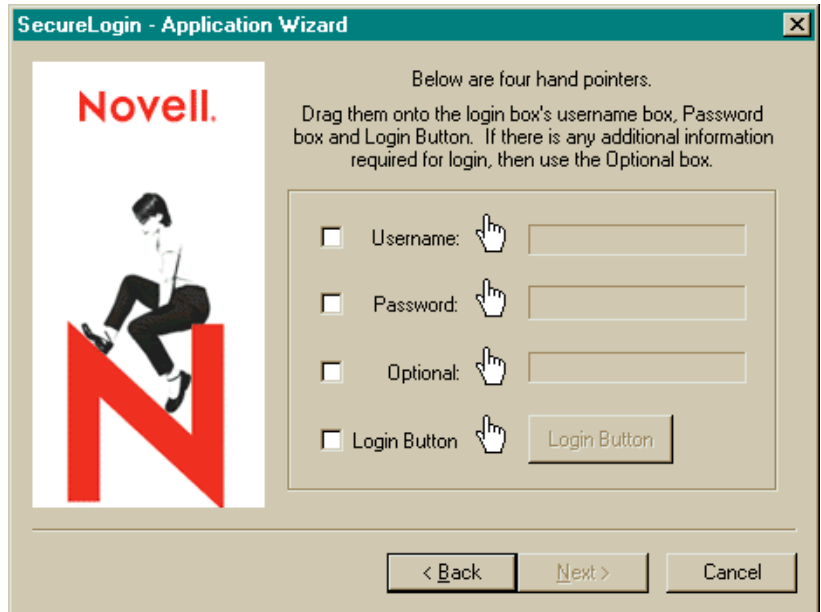
To view the newly added script:

- 1** Click Login Details.
- 2** Double-click the application listed in the platform column.

Setting Up a Windows Application

- 1** At the Novell SecureLogin tool's main screen, select Login Wizard.
- 2** Enter a description, for example, Remedy.
- 3** Select Windows Application > click Next.
- 4** Follow the on-screen prompts.

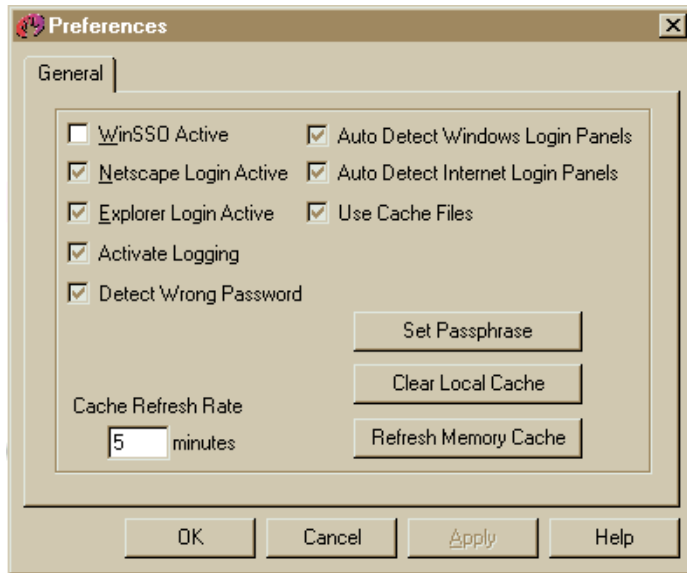
You drag and drop hand icons into login fields.



Setting Preferences

Use the Preferences option to customize the operation of SecureLogin. You can restrict users from accessing the Preferences option.

The following figure illustrates preferences that you can set.



WinSSO Active

To enable the Windows single sign-on features of SecureLogin, check the WinSSO Active check box. To disable Windows single sign-on, uncheck the check box.

Default: On (checked)

Netscape Login Active

To enable the Netscape single sign-on features of SecureLogin, check the Netscape Login Active check box. To disable Netscape single sign-on, uncheck the check box.

Default: On (checked)

Explorer Login Active

To enable Microsoft Internet Explorer features of SecureLogin, check the Explorer Login Active check box. To disable Internet Explorer single sign-on, uncheck the check box.

Default: On (checked)

Activate Logging

To log the details of use to the hard drive, check the Activate Logging check box. However, because this preference is used for debugging and troubleshooting, do not use this option unless Technical Services advises you to. Leave the check box unchecked.

Default: Off (unchecked)

Detect Wrong Passwords

To enable SecureLogin to attempt to detect whether you have given it an incorrect password, check the Detect Wrong Passwords check box. SecureLogin then prompts you to change the password.

Default: On (checked)

Auto Detect Windows Login Panels

The Auto Detect Windows Login Panels setting controls whether the Windows single sign-on component automatically detects Windows login panels. To receive a prompt to run the wizard, check the check box.

Default: On (checked)

Auto Detect Internet Login Panels

The Auto Detect Internet Login Panels setting controls whether the Web single sign-on component automatically detects Web login panels. To receive a prompt to run the wizard, check the check box.

Default: On (checked)

Use Cache Files

Username and passwords are normally stored in a directory on the server, but if the server is unavailable, or if you are using a notebook computer, the cache is used. The cache is password protected and encrypted.

To enable SecureLogin to use cache files, check the Use Cache Files check box.

Default: On (checked)

Cache Refresh Rate

The cache refresh rate controls the number of minutes that SecureLogin waits between synchronizing the information between eDirectory and the local cache.

Default: 2 (minutes)

Additional Controls

Set Passphrase

The Set Passphrase option enables you to reset your selected passphrase and password combination.

- 1 Click Set Passphrase.
- 2 Enter the password to your passphrase > click OK.
- 3 Enter a new passphrase question and answer.
- 4 Confirm the answer > click OK.

Clear Local Cache

To clear the entries held in the local cache, click Clear Local Cache.

Refresh Memory Cache

To force SecureLogin to immediately synchronize the data between NDS[®] eDirectory and the local cache, click Refresh Memory Cache.

Disabling Three Preferences

The Active setting on the administrative tool controls whether the following preferences are enabled:

- ♦ WinSSO Active
- ♦ Netscape Login Active
- ♦ Explorer Login Active

By default, the Active button is checked. This setting enables the three preferences. To disable these preferences, uncheck the Active button.

4 Administering Scripts

Types of Scripts

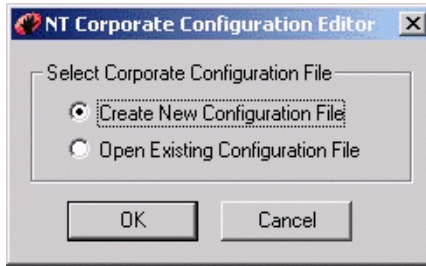
The following table lists types of scripts that SecureLogin supports.

Type of Script	Description
Windows* Applications	For Windows-based applications
Terminal Launcher	For applications that require access via an emulator
Internet Page	For Web-based applications
Password Policy	For password policy associated with one or many applications
SecureLogin Startup	For the execution of an application during the startup of SecureLogin
Corporate Scripts	Corporate scripts are indicated on the client by the letter C next to the icon for that script

Administering Scripts for NT

You can administer corporate scripts in an NT environment.

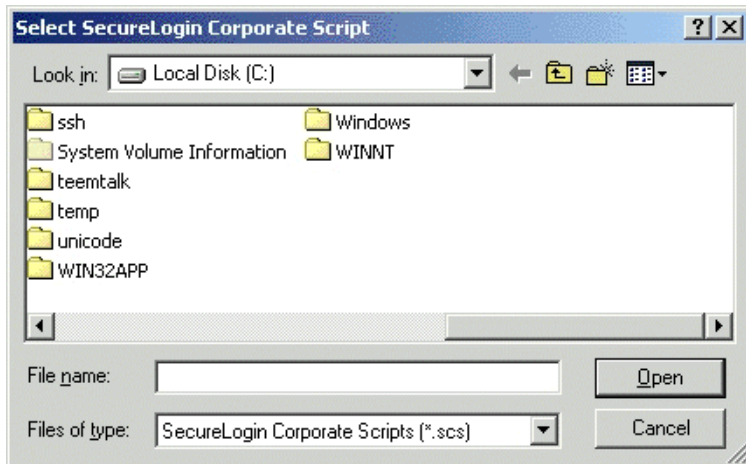
- 1 Run the NT Corporate Script Editor from the Start menu.
- 2 From the NT Corporate Configuration Editor screen, select an option.



Create New Configuration File enables you to create a new corporate script for your environment.

Open Existing Corporate Script enables you to modify an existing corporate script for your environment.

- 3 Select a current corporate script or define a name for a new corporate script.



The location of this corporate script must be in a shared directory that users have access to.

Access to the corporate script files is governed by the standard NT file permissions.

IMPORTANT: Ensure that general users have Read Only access to this file.

- 4 Select a configuration to edit.

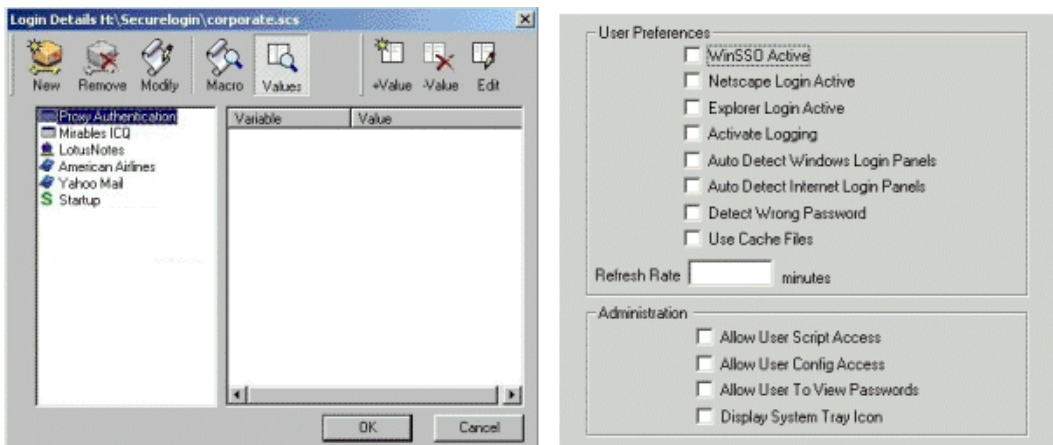


To edit the current corporate scripts, select Edit Scripts.

To edit the current preferences for this corporate script, select Edit Preferences.

To browse for another script to edit, select Edit Another Script.

The following figure illustrates the main NT script editor window and the preferences page for the current corporate script.



This editor operates in the same manner as the user script editor. However, no user variables are stored at this location and there is no Display Passwords option.

For more information refer to [“Managing SecureLogin” on page 37](#).

A user's configuration can be set up to load a corporate script in one of two ways:

- ◆ Pass in the path to the script as the argument to PROTO.EXE.

Edit the registry and change the setting HK_CURRENT_USER/Software/Microsoft/Windows/Current/Run so it has the corporate script location after PROTO.EXE.

- ◆ Remove the above key and create a shortcut similar to the following:

```
C:\PROGRAM FILES\SECURELOGIN\PROTO.EXE
S:SECURELOGIN\PRODUCTION.SCS
```

Place this shortcut in the user's startup folder.

To ensure that SecureLogin will also read from this location, another registry setting exists to keep track of the last location a corporate script was run from. The location for this registry setting is HKEY_CURRENT_USER\software\Protocom\SecureLogin\LastNTCORPLocation.

If you close and reload SecureLogin, it will load the corporate script from the last known location.

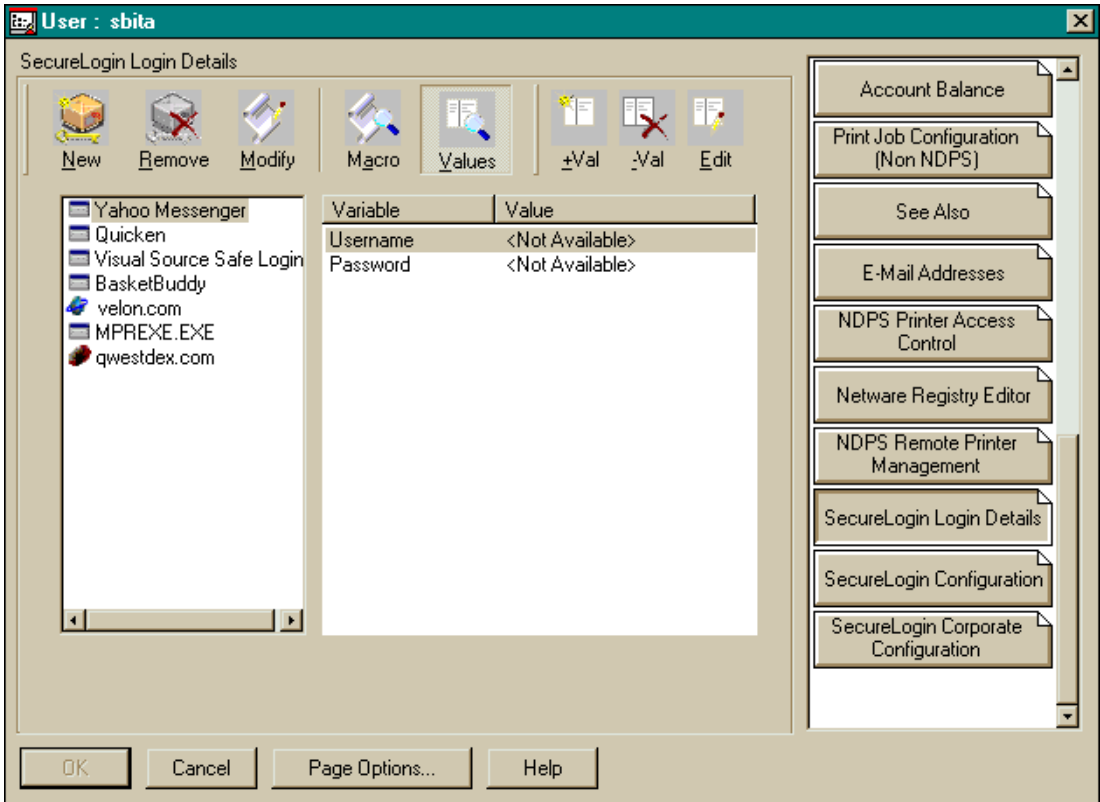
Administering Scripts for eDirectory

The SecureLogin snap-ins for NetWare Administrator and ConsoleOne currently work with NWADMN.EXE, NWADMIN95.EXE, and NWADMIN32.EXE. These snap-ins allow you to set up corporate scripts as well as set up user configurations.

When installed, these snap-ins add two new pages to the administration details of User, Organizational Unit, and Organization objects.

Using the SecureLogin Details Tab

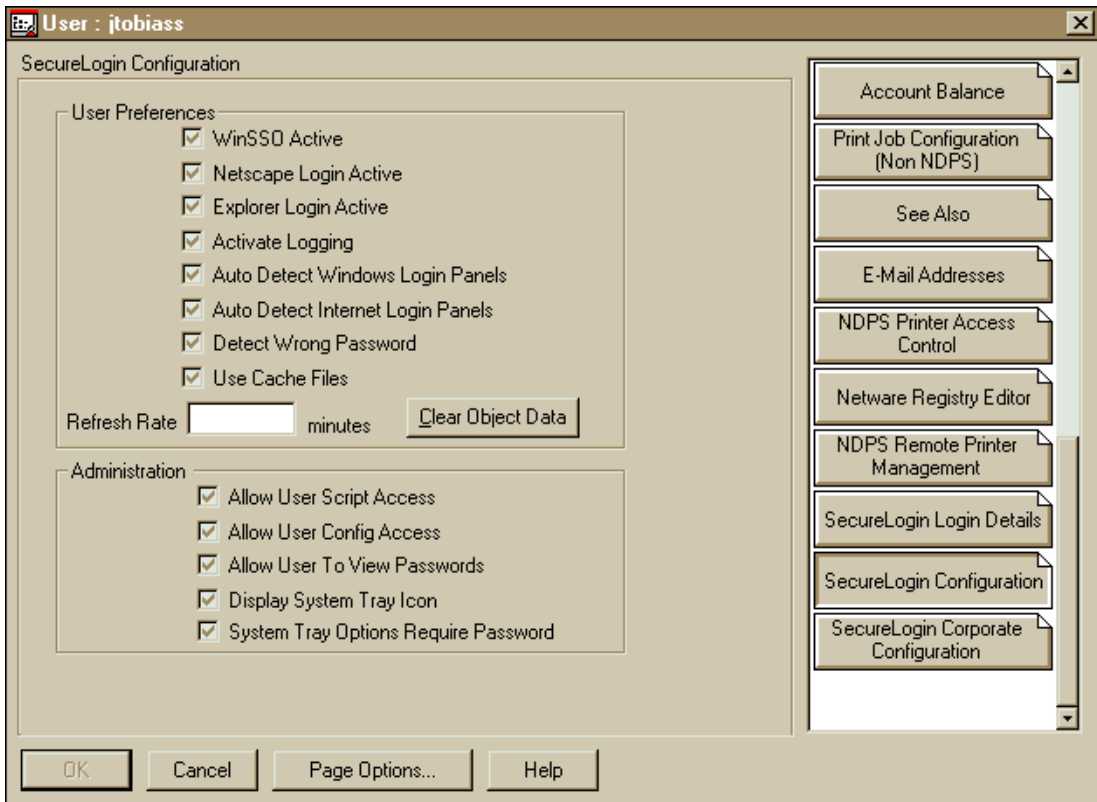
This page operates the same way as the script editor tool as described in [Editing Login Details](#). The only difference is that in the NetWare Administrator version a password variable cannot be viewed. This omission prevents anyone from stealing passwords.



On Organizations and Organizational Units, this page is used to write corporate scripts. Corporate scripts can be created and entered manually or copied from a wizard-generated example.

Using the SecureLogin Configuration Tab

This page is similar to the preferences page for workstations. However, several extra parameters are available only through the snap-ins to NetWare Administrator and ConsoleOne™. These parameters enable you to limit users' access to certain features.



Preventing User Access to the Script Editor

To prevent users from using the New, Modify, and Macro buttons in the script editor, uncheck Allow User Script Access.

Default: On (checked)

Preventing Users from Setting Preferences

To prevent users from using the Preferences option on the SecureLogin administration tool, uncheck Allow User Config Access.

Default: On (checked)

Preventing Users from Viewing Passwords

To prevent users from viewing their own SSO passwords, uncheck Allow User to View Passwords.

Default: On (checked)

Displaying the System Tray Icon

To prevent users from displaying and accessing the system tray icon, uncheck Display System Tray Icon.

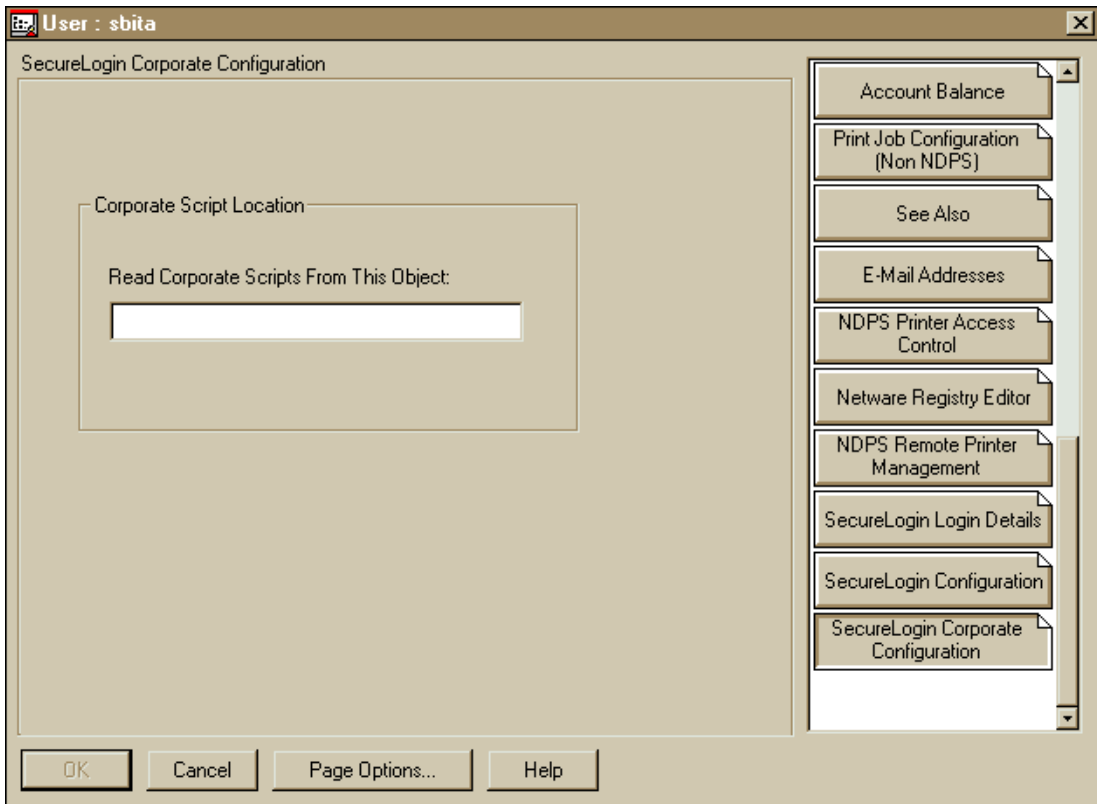
Default: On (checked)

Requiring a Password for the System Tray

If you set this option, users must provide their passwords before they can access options on the system tray's icon.

Using the SecureLogin Corporate Configuration Tab

This page enables you to select where SecureLogin reads its corporate configuration and application information from. By default, SecureLogin reads its information from the current user's context and then searches up the eDirectory™ tree. This setting allows you to prevent upward searches.

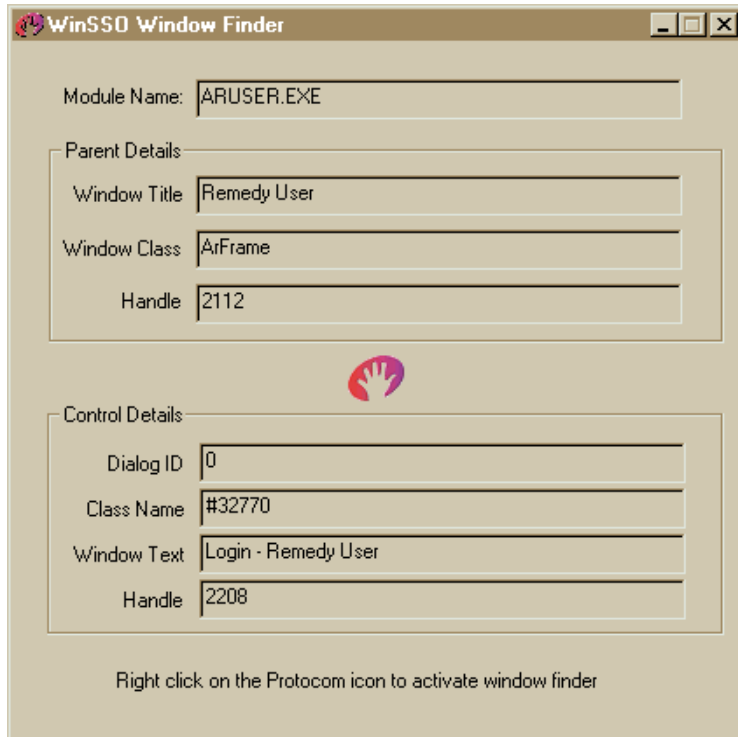


A User object or Container object will read its corporate scripts. Configuration and application information can be read from a specific context within eDirectory by pointing this configuration page directly to it.

Finding Dialog Control IDs

Most of the script commands related to logging into Window applications require a dialog control ID. A control ID is a number that uniquely identifies a field (called a control) within a window.

To enable you to efficiently determine these control IDs, SecureLogin includes a tool on the start menu, called the Windows Finder.



To inspect a control, right-click the SecureLogin icon and drag it over the control of interest. The Windows tool displays the details of the control.

If an application page hides the Window Finder, click the WinSSO Window Finder icon on the system tray.

Module Name: The name of the executable that created the window.

Use this name for the platform name of the Windows single sign-on script.

Window Title: The title of the window that contains the control.

You can use this title in a window or title statement.

Window Class: A field for information only.

Each window has a class associated with it.

Dialog ID: A unique identifier.

Each control has a unique identifier called the control ID. Use this number as the target for Type, Click, Ctrl, and Setplat statements. For information on each of these commands, see [Chapter 7, “SecureLogin Commands,” on page 77](#).

Class Name: A name determines the type of the control.

For single sign-on to work correctly, the SecureLogin Windows component must be able to read and write text to the specified control. The class name determines the type of the control and whether reading and writing is possible. Supported classes include edit, combobox, and static.

Window Text: A field displays the text contained within the control.

This information can be useful in troubleshooting and for writing the regular expression required by the Setplat command.

For more complicated applications, advanced diagnostic tools such as Window Watcher are available from Technical Services.

Predefined Applications

SecureLogin provides native script support for many popular applications so that you don't have to configure them manually.

Application	Application	Application
America Online*	ImageRight	ICQ
ACT Contact Manager	Siebel* Customer Tracking	Cerner
AOL Instant Messenger*	Visual SourceSafe	CICS*
ACF2*	Yahoo!* Messenger	Lotus Organizer*
Centra Symposium*	Clarify	meetingmaker*
CorporateTime	Solaris* Login	Microsoft* Internet Gaming Zone
Entrust* Client	AIX* Login	Microsoft Front Page
Entrust Server	Linux* Login	Microsoft* Money 98/99
Eudora* Email	HP* Login	MSN Messenger
GoldMine*	PC/Support	QuickBooks Pro*
Soffront	MMIS	Quicken*
PeopleSoft*	Oracle*	Remedy* ARUSER

Application	Application	Application
AS/400*	Oracle Financials	Remedy Notifier
MS SQL	Mobile Up	GroupWise®
RiskMaster	Lawson	cc:Mail*
PlusW33	RACF	Lotus Notes*
SuperSession	TSO	IBM* Memo
ACF/2	Citrix*	Top Secret
SAP/R3	Model 204	

5

Setting Up Terminal Emulation

Planning for Microsoft Terminal Server and Citrix MetaFrame

SecureLogin supports Microsoft* Terminal Server and Citrix* Corporation MetaFrame remote application environments.

When using SecureLogin with a remote desktop, no special configuration is required in either environment, but when planning for single sign-on you need to consider the following:

- ◆ Where the application is being run from.

To enable applications that are run from the server to use single sign-on functionality, install SecureLogin on the Terminal or MetaFrame server. Terminal Server and MetaFrame only send keystrokes and screen images to the PC (workstation). This process means that SecureLogin installed on the local computer is unable to determine which applications are running remotely.

When SecureLogin is installed on Terminal Server, SecureLogin can see the applications' Windows and Terminal information natively and thus perform single sign-on.

- ◆ Whether single sign-on through the Initial Citrix/Terminal Server GINA is required.

SecureLogin includes a Citrix/Terminal Server module. This module automates the initial login to the Citrix/Terminal Server GINA by providing the user's details from the local machine. This process requires the installation of a login extension on both the client and the Terminal and MetaFrame servers.

Using SecureLogin Terminal Launcher

SecureLogin can provide single sign-on from a user's workstation to back-end mainframe and UNIX* systems. To achieve this, SecureLogin needs to be plugged in to the emulator that the user uses to connect to this back-end system. Terminal Launcher (TLAUNCH.EXE) provides this connectivity.

Terminal Launcher does the following:

- ◆ Acts as the translator between the SecureLogin login sequence and a user's specific variables (for example, the username and password)
- ◆ Coordinates the information being entered onto a mainframe or Telnet screen

You configure Terminal Launcher so that Terminal Launcher connects to the correct mainframe or Telnet emulators and waits for the right login sequence before entering usernames and passwords.

The Terminal Launcher can be configured to launch the terminal session and then run a script to log the user in to the system. Further, Terminal Launcher can perform any keystrokes within an application that a user can do manually.

At the corporate level, the same script can be used for all who log in. Only the username and password and other login-specific variables change.

Configuring an Emulator

SecureLogin supports the following emulator types.

Emulator Type	Emulator
HLLAPI	Eicon* Aviva Attachmate* Extra Jolly Giant QWS3270 Plus* IBM* Personal Communications Wall Data RUMBA* WRQ Reflection* 7 for IBM WRQ Reflection 8 for IBM Netmanage Chameleon* Hostlink Netmanage NS/Elite Hummingbird* HostExplorer* Pericom teemtalk*
DDE	NCP3270

Emulator Type	Emulator
VBA	WRQ Reflection 5.21 for UNIX and Digital WRQ Reflection 7 for UNIX and Digital WRQ Reflection 8 for UNIX and Digital
Generic	Context Microsoft* Telnet for Win95/98 Microsoft Telnet for WinNT/2000
Other	Tera Term Pro

To configure an emulator, complete the following steps.

- 1** Install the mainframe or Telnet emulators that the user will be connecting with.
- 2** Identify the mainframe login sequence or prompts (for example, Login).
- 3** Configure TLAUNCH.EXE to know about the emulators you wish to use for single sign-on.

You can define as many emulators as you would like to.

- 3a** Launch TLAUNCH.EXE by clicking Start > Programs > SecureLogin > Terminal Launcher.
- 3b** Click Edit Available Emulators.

A screen displays a list of emulators that have been pre-configured. To suit your environment, you can add, edit, or delete emulators.

Terminal Launcher stores its settings in the TLAUNCH.INI file. As installed, this file contains information about a list of emulators that have been tested with SecureLogin. For configuration documents on additional emulators, contact [Technical Services \(http://www.support.novell.com/\)](http://www.support.novell.com/).

You can use these pre-configured emulators as examples. To configure an emulator, you must specify to the emulator the correct path for the executable and the mainframe session file (if one is required).

- 4** Write the login script to perform the sign-in.
- 5** Create a desktop icon for the Terminal Launcher to replace the mainframe emulator icon, or configure a background connection mode if your emulator supports such a mode.

To get started, launch the TLAUNCH.EXE program from the Start, Programs, SecureLogin, Terminal Launcher option. Click Edit Available Emulators in the middle of the terminal launcher screen. This displays a screen that contains a list of the emulators that have been pre-configured and may be removed or edited to suit your environment.

Configuring an Emulator Script

The following example explains how to create a script, configure the login, and set up a shortcut for the application, using Eicon* Aviva as the emulator. This process enables you to access the session by clicking an icon on your desktop.

Creating a Script

- 1** Right-click the SecureLogin icon on the system tray > select Login Details > click New.
- 2** Enter a name (for example, Simple Login) in the Platform Name box > click OK.
- 3** Enter the script.

For example, enter the following:

```
WAITFORTEXT "Enter USERID"  
TYPE $username  
TYPE @E  
WAITFORTEXT "password"  
TYPE $password  
TYPE @E
```

- 4** Save the script by clicking OK.

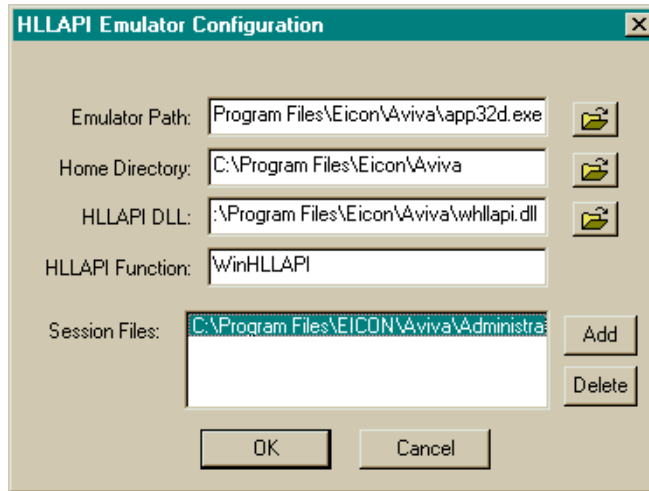
Configuring the Login

- 1** Launch Terminal Launcher.
- 2** Select Simple Login from the list of available applications < click Add > OK > Done.
- 3** Configure Eicon Aviva to use Simple Login by clicking Edit Available Emulators > Eicon Aviva > Edit.

If an emulator is not in the list, add the emulator. Click New > enter a name > select an emulator type > click OK.

- 4** Fill in the fields with their correct values > click OK.

The following figure illustrates these fields.



5 Click OK > Done.

6 Ensure that Eicon Aviva is selected from the drop-down list in the Emulator box.

This box is in the bottom left corner of the Terminal Launcher screen.

NOTE: If you select OK instead of Close, Terminal Launcher runs the selected script (application) after launching Eicon Aviva. However, when you select Close, the program saves this selection to USERSETTINGS.INI, a file that allows SecureLogin to run the script from a command line.

7 At Options, check the Save Settings on Exit check box > click Close.

Setting Up a Shortcut

You can set up a shortcut for the application.

1 Right-click the desktop > select New > Shortcut.

2 Enter (or browse to) the path for TLAUNCH.EXE.

For example, enter “C:\PROGRAM FILES\SECURELOGIN\TLAUNCH.EXE”. Include quotation marks.

3 To the end of this line, add `"/auto`.

This addition enables the command line facility of Terminal Launcher. Include the quotation mark. An ending quotation mark comes at the end of Step 5.

4 Add */pname_of_application*

For example, add `/pSimple Login`.

NOTE: Enter the application name exactly as it appears in the Available Applications list in Terminal Launcher.

5 Add */ename_of_emulator*.

For example, add `/eEicon Aviva`.

NOTE: Enter the emulator name exactly as it appears in the Available Emulators list in Terminal Launcher.

The box contains this shortcut line:

```
"C:\ProgramFiles\Novell\Securelogin\Tlaunch.exe"  
/auto/pSimple Login /eEicon Aviva
```

6 Click Next > name the new application shortcut > click Finish.

When you double-click the shortcut, the shortcut launches Eicon Aviva and runs the selected script.

The SecureLogin Terminal Launcher can launch up to 15 applications at a time, as long as you have enough sessions defined for the particular emulator you are using. To open several applications at once, add one more */panother_application_name* for each additional application.

Using Command Line Parameters

Terminal Launcher can use the following command line parameters.

Parameter	Description
<i>/auto</i>	Tells Terminal Launcher that you are running the command line version. This parameter must be in the command line for the other command line options to work.
<i>/b</i>	Specifies background authentication mode.
<i>/emulator_name</i>	Launches the specified emulator.

Parameter	Description
<i>/hllapi_short_name</i>	Forces Terminal Launcher to connect to the specified HLLAPI session.
<i>/kexecutable_name</i>	Kills the specified executable before launching an emulator.
<i>/m</i>	Allows multiple (sequential) connections to particular sessions.
<i>/n</i>	Launches the selected emulator without running a script (equivalent to the Emulator Only check box in the main program).
<i>/nnumber_1-15</i>	Launches the specified number of sessions without running scripts.
<i>/papplication_name</i>	Runs the specified application.
<i>/q</i>	Specifies Quiet Mode (no cancel dialog).
<i>/s</i>	Suppresses errors.
<i>/t</i>	Enables unlimited timeout when connecting to an emulator.

The following examples include parameters that Terminal Launcher uses:

- ◆ Tlaunch.exe /auto /eEicon Aviva /pApplication1 /pApplication2
- ◆ Tlaunch.exe /auto /pTSO
- ◆ Tlaunch.exe /auto /n3
- ◆ Tlaunch.exe /auto /q
- ◆ Tlaunch.exe /auto /eEicon Aviva /pBackground /b /t /m /hA /s /q

When an emulator or application is not specified on the command line, Terminal Launcher uses the settings stored in the user settings file (USERSETTINGS.INI). You can modify these settings.

- 1** Check the Save Settings on Exit check box.
- 2** Close the main program.

Configuring Backup Sessions

Each Terminal Emulator that is configured must have a number of backup sessions configured for it. For most emulators, you are required to have one session file for every session that you want to have running at the same time. These are usually stored as separate files.

When you configure an emulator for use with Terminal Launcher, you must input a session file for it to use. To tell Terminal Launcher that it can use more than one session file, complete the following steps:

- 1 Launch Terminal Launcher > click Edit Available Emulators.
- 2 Select the correct emulator from the Available Emulators window > click Edit.
- 3 Add the backup session files to the Session Files dialog box.

You can launch only as many emulator sessions as there are session files defined.

NOTE: After the emulator is launched, these session files will be executed as a command line parameter. Some emulators (such as QWS3270 Plus) do not have session files. Instead, these emulators have individual sessions stored in the registry. Think of these session files as command line parameters that will be passed to the executable.

Using Terminal Launcher With Non-HLLAPI Compliant Emulators

You can use Terminal Launcher with Terminal Emulators that do not support HLLAPI but do support scripting that is able to call external DLL files. To do this, you must create a script that asks SecureLogin for commands one at a time and then interprets the commands received.

The following script has been tested with Reflection* 8 for UNIX and Digital*.

```
Sub SecureLogin()  
  Dim SecureLoginObject As ISLBroker  
  Dim ReturnCode As Long  
  Dim Data As String  
  Dim targ As Long  
  Dim FunctType As Long  
  Dim CR As String  
  Dim temp As String  
  
  Session.Wait 0.1'The waits are necessary for the screen to  
  be updated.
```

```

Set SecureLoginObject = New SLBroker
CR = Chr$(rcCR) ' Chr$(rcCR) = Chr$(13) = Control-M
SecureLoginObject.LoadScript
While (1 = 1)
    FunctType = 0
    'retrieve command from VBABork
    SecureLoginObject.GetCommand FunctType, targ, Data
    If FunctType = SecureLoginObject.SetCursor Then
        ' SetCursor is not supported
        ReturnCode = 0
    ElseIf FunctType = SecureLoginObject.TypeText Then
        If (StrComp(Data, "@E", vbTextCompare) = 0) Then
            Session.Transmit CR
        Else
            Session.Transmit Data
        End If
        Session.Wait 0.1
        ReturnCode = 0
    ElseIf FunctType = SecureLoginObject.ScanForText Then
        bResult = Session.FindText(Data, Session.ScreenTopRow, 0)
        ReturnCode = 0
        If bResult = True Then
            ReturnCode = 1
        End If
    Else
        ' End of script
        GoTo ErrorHandler
    End If
    SecureLoginObject.SetReturnCode ReturnCode
Wend
ErrorHandler:
End Sub

```

The script should also work for Reflection 7. Reflection 6 and earlier versions require a different Reflection script because these versions use Reflection Basic instead of VBA (Visual Basic* for Applications). If you require a script for Reflection 5.21, contact [Novell® Technical Services \(http://www.support.novell.com/\)](http://www.support.novell.com/).

To use Reflection for UNIX and Digital, you must add this macro (the Sub SecureLogin() script) by using the macro editor in Reflection. In addition to adding this macro, you must go to the macro editor, select Tools > References, and check the check box titled vbabork2 1.0 Type Library.

If this option is not displayed, ensure that VBABORK2.DLL exists in the SecureLogin directory. If it does not, re-install SecureLogin, making sure to select Terminal Launcher.

If the vbabork2 1.0 Type Library option displays, you must register it.

- 1** Open a DOS shell.
- 2** Enter **regsvr32** followed by the path to VBABORK2.DLL.

For example, enter

```
regsvr32 "C:\Program Files\SecureLogin\  
vbabork2.dll"
```

A message should indicate that DllRegisterServer succeeded.

- 3** Add the reference to this module in Reflection as described above.

Only one session may be launched at a time using SecureLogin. To run the script, you must run the SecureLogin macro once the session has been opened. This may be done automatically in Reflection by selecting Connect Macro from the Connection Setup menu.

By doing this procedure, the SecureLogin macro will run every time the session is opened. Without this procedure, you will need to manually run the macro when you want to run the script.

To run the script, you must set up the emulator in Terminal Launcher.

- 1** Launch Terminal Launcher > click Edit Available Emulators. Set up the emulators per normal, but set the HLLAPI type to None.
- 2** Select the emulator.
- 3** Enter anything in the HLLAPI DLL and HLLAPI Function boxes.
- 4** Click OK > Done.

Determining Which Session File To Use Automatically

A session file tells the emulator how to connect to the mainframe. In some environments, and with emulators like Attachmate Extra, users on the network may have named their session file uniquely. This means that Terminal Launcher may need to be configured individually for each user.

Terminal Launcher includes a special option that allows it to determine the last-used session file and start that mainframe configuration. This option reduces or eliminates the need to manually configure each user's environment for these type of emulators.

For example, when using Attachmate* Extra's 3270 mainframe emulator, the user starts the emulator, specifying the emulator's configuration session file. Depending on your corporate configuration, the session configuration filename may be different for each user.

When SecureLogin is run in direct mode, SecureLogin launches the emulator on the user's behalf, with the user's own configuration settings. For this to occur, the Terminal Launcher needs to be told what the correct configuration file is. SecureLogin's terminal launcher has a feature that allows it to search for the last used mainframe configuration file and then start the emulator by using this file.

To configure Terminal Launcher to use the last-used configuration file, the command %Latest is used in the Session Files section of Terminal Launcher.

For example, to launch the latest configuration of the Attachmate Extra mainframe, you could enter the following in the Session Files section:

```
%Latest C:\Program Files\E!98\Sessions\*.edp
```

This entry causes Terminal Launcher to search the E!98\Sessions directory, looking for the .EDP file with the newest date and time. Terminal Launcher then launches that file with the emulator and connects to the mainframe.

If the file MAINFRAME.EDP had the most recent date and time in the Sessions directory, the resultant command line would look like the following:

```
C:\Program Files\E!98\extra.exe c:\Program  
Files\E!98\Sessions\mainframe.edp
```


6

Using Password Policies

SecureLogin has a powerful facility that allows you to ensure that variables comply with certain rules governing their composition. Although this feature is called “password policies,” these policies may be used on any variables, not just password variables.

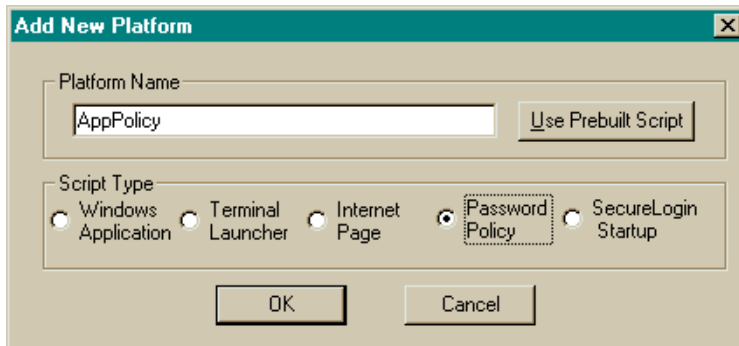
To restrict a variable to a policy:

- ◆ Create a password policy script.
- ◆ In the script where the variable you wish to restrict exists, set that variable to be restricted by the policy you have created.

Creating a Password Policy

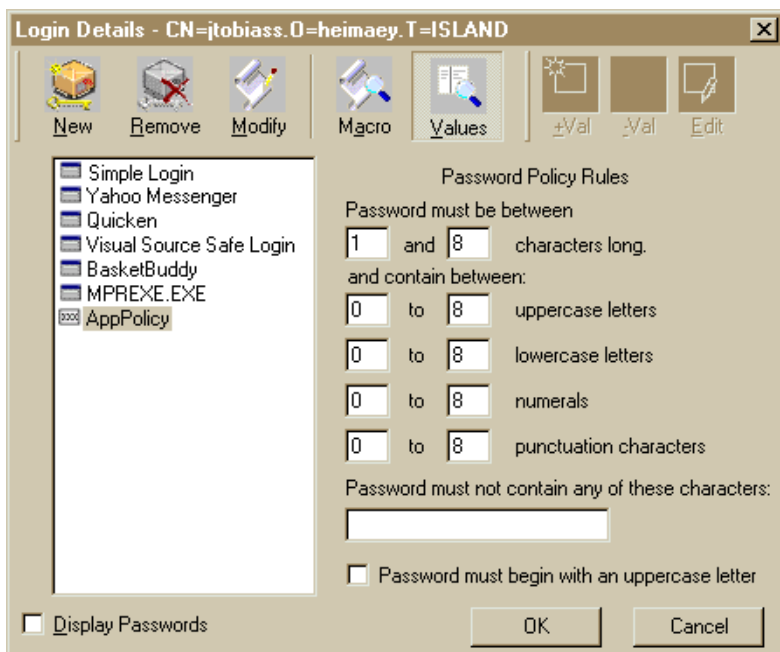
To create a password policy script, create a new script by using the user administration tool, NetWare® Administrator, or ConsoleOne™. The following procedure is for the user administration tool.

- 1** Right-click the SecureLogin icon on the system tray > select Login Details.
- 2** Click New > enter a name for the script > select Password Policy as the script type > click OK.



- From the list of platforms, select the new script you just created.

Notice that instead of the usual list of variables on the right side, the administration tool contains a number of fields that contain values pertaining to the password policy. The following figure illustrates these fields.



- Set rules for the password policy.

The values allow you to set certain criteria or rules. For example, the password must be a certain length and it cannot contain certain characters

Setting Advanced Criteria

In addition to the restraints under Password Policy Rules, you can set advanced criteria.

- 1 From the list of platforms on the Login Details screen, select the password policy script (platform) that you wish to edit > click Modify.

You can also double-click the platform.

- 2 Enter the commands into the password policy script.

Advanced Command	Description
NODUPLICATECHARACTERS <i>“caseinsensitive”</i>	Means that a character will not appear more than once in a password.
NOREPEATINGCHARACTERS <i>“caseinsensitive”</i>	Means that a character will not be the same as the preceding character.
NOSEQUENTIALCHARACTERS <i>“caseinsensitive”</i>	Means that a character will not be one character above or below the preceding character.
POSITIONCHARACTER <i>character set positions</i>	Means that a character from the specified character set must exist in at least one of the specified positions. The <i>character set</i> may be uppercase, lowercase, punctuation, or numeral. This parameter must have a character position listed by number. If multiple positions exist, they must be separated by commas.

The following is an example of the POSITIONCHARACTER command:

```
POSITIONCHARACTER NUMERAL 3,4,5
```

This command ensures that a numeral exists in a password in at least one of positions 3, 4 or 5. The password b{0vf4 would pass this test because the numeral 0 exists in position 3.

You are able to have multiple instances of this command in a password policy. If you wanted to ensure that a password had a numeral in positions 3, 4 and 5, you would need to have the following three commands:

```
POSITIONCHARACTER NUMERAL 3
POSITIONCHARACTER NUMERAL 4
POSITIONCHARACTER NUMERAL 5
```

You must add these four advanced commands manually by editing the script. When doing so, you will see the other commands which can be edited by using the normal password policy screen. It is recommended that you edit these values using the normal edit screen.

Using a Password Policy Script

You can restrict a variable to a particular password policy script.

- 1** Select the script that contains the variable you want to restrict.
- 2** Add the following line at the top of the script:

```
RESTRICTVARIABLE variable_name password_policy
```

To restrict multiple variables, you just need to add multiple RestrictVariable commands.

The *variable_name* parameter can be a normal variable (for example, \$Password) or a runtime variable (for example, ?temp). This flexibility can be useful if you change a password by using a runtime variable and then set a normal variable to the value of the runtime variable.

Adding the RestrictVariable command is all you need to do to ensure that a variable will comply with the policy. The value entered will be rejected if it does not comply with the policy set for that variable, regardless of whether the variable is being added or changed through SecureLogin or through a script that is running.

If the value being changed by a user is not accepted, a message informs the user as to why the value wasn't permitted. If the value is being set through the ChangePassword command being run in automatic (random) mode, the value generated will comply with the policy.

In some cases, a policy may be created where no acceptable values exist. When this occurs, an error will be displayed when the ChangePassword command tries to generate a password.

For more information on the RestrictVariable command, see [“Script Reference Guide” on page 118](#).

Values will not be forced to comply with password policies if you use the SecureLogin SET command to set them.

Example Password Policy Scripts

Example 1

```
MAXPASSWORDLENGTH 8
MINPASSWORDLENGTH 8
MAXPUNCTUATION 0
MINPUNCTUATION 0
MAXUPPERCASE 8
MINUPPERCASE 0
MAXLOWERCASE 8
MINLOWERCASE 0
MAXNUMERALS 8
MINNUMERALS 0
```

This password policy indicates that the password must be exactly 8 characters long and contain no punctuation characters.

The password `asdf4jB8` is acceptable.

The password `aasdf5$n` is unacceptable because it contains a punctuation character.

Example 2

```
MAXPASSWORDLENGTH 16
MINPASSWORDLENGTH 6
MAXPUNCTUATION 8
MINPUNCTUATION 0
MAXUPPERCASE 16
MINUPPERCASE 1
MAXLOWERCASE 16
MINLOWERCASE 0
MAXNUMERALS 16
MINNUMERALS 0
BEGINWITHUPPERCASE
DISALLOWEDCHARACTERS @&
```

This password policy indicates that the password must be between 6 and 16 characters long. It must contain at least one uppercase character. It can contain no more than 8 punctuation characters. It must begin with an uppercase character, and it may not contain either the `@` character or the `&` character.

The password `R48iv"?'` is acceptable.

The password `R48?-` is unacceptable because it is less than 6 characters long.

Example 3

```
MAXPASSWORDLENGTH 12
MINPASSWORDLENGTH 6
MAXPUNCTUATION 8
MINPUNCTUATION 0
MAXUPPERCASE 8
MINUPPERCASE 0
MAXLOWERCASE 8
MINLOWERCASE 0
MAXNUMERALS 8
MINNUMERALS 0
NODUPLICATECHARACTERS CASEINSENSITIVE
POSITIONCHARACTER NUMERAL 3,4,5
```

This password policy indicates that the password must be between 6 and 12 characters long. It can contain no more than 8 of any character type (uppercase, lowercase, numeral, or punctuation). No character may appear more than once in the password, regardless of case. A numeral must appear in at least one of positions 3, 4, or 5.

The password f54v9)_Q is acceptable.

The password f5v)_QF7 is unacceptable because it has no numeral in positions 3, 4 or 5, and the letter F occurs in positions 1 and 7.

7

SecureLogin Commands

This section lists and explains commands that you can use in SecureLogin scripts.

Commands

- “ChangePassword” on page 78
- “Class” on page 80
- “Click” on page 81
- “Ctrl” on page 82
- “Delay” on page 82
- “Dialog / EndDialog” on page 83
- “DisplayVariables” on page 84
- “EndScript” on page 85
- “If / Else /EndIf” on page 86
- “IfText / EndIfText” on page 88
- “Increment / Decrement” on page 89
- “KillApp” on page 90
- “MessageBox” on page 91
- “Parent / EndParent” on page 92
- “PickListAdd” on page 94

“PickListDisplay” on page 95

“ReadText” on page 96

“RegSplit” on page 97

“Repeat” on page 98

“RestrictVariable” on page 99

“RestrictVariable” on page 99

“Run” on page 100

“Strcat” on page 101

“Set” on page 102

“SetCursor” on page 103

“SetFocus” on page 103

“SetPlat” on page 104

“Setprompt” on page 107

“Title” on page 108

“Type” on page 108

“WaitForFocus” on page 110

“Waitfortext” on page 111

ChangePassword

Usage: CHANGEPASSWORD *variable* [random]

Used in: Windows*, Terminal Launcher.

Command Type: Action

Arguments:

variable - A normal or run time variable that the new password is stored in.

[random] - If specified, invokes the random password generator.

Remarks:

The ChangePassword command allows for a single variable to be changed. Use this command in scenarios where password expiration is an issue. The variable passed in will be set to the new password.

The flag for this command is “random.” If the “random” flag is set, the user will not be prompted and the password will be generated automatically, based on the platform's password policy.

If the “randomness flag is not selected, SecureLogin displays a dialog box, prompting the user to enter a new variable. SecureLogin then ensures that the new variable complies with any password policies that may apply.

For more information on password policies, see [“RestrictVariable” on page 99](#).

Example Terminal Launcher Script

```
restrictvariable $Password "password policy script"
waitfortext "Enter USERID"
type $Username
type @E
waitfortext "Enter Password"
type $Password
type @E
repeat 100
iftext "User logged in successfully"
break
endiftext
iftext "Password expired"
changepassword $Password random
type @T
type $Password
type @T
type $Password
type @E
break
endiftext
delay 50
endrepeat
```

Class

Usage: CLASS *window-class*

Used in: Windows

Command Type: Dialog specifier

Arguments:

window-class- A string specifying the window class that this statement will match.

Remarks:

When a window is created, it is based on a template known as a window class. The CLASS statement checks to see if the class of the newly created window matches its <window-class> argument. If so, execution continues on the next line. Otherwise, the dialog statement does not match and execution continues at the next dialog statement.

The class can be determined by using Window Finder. See [“Finding Dialog Control IDs” on page 54](#).

Example Windows Script

```
Dialog
class "Button"
title "&OK"
Parent
Title "Password Verification"
Parent
Title "XXXXXX"
class "#32770"
EndParent
type "$Password" #1159 password
EndParent
EndDialog
type "\N "

repeat 100
if ?count eq 15
type -raw "hello world"
set ?count <NOTSET>
endscript
endif
increment ?count
delay 40
endrepeat
```


Click

Usage: CLICK *ctrl-ID flag x y*

Used in: Windows

Command Type: Action

Arguments:

ctrl-ID - The dialog ID of the button to be clicked.

[-raw] - Eliminates the mouse and sends a direct click.

Remarks:

The Click command sends a click command to the button passed in as the argument. If the button does not return a #ID, use Type “n”.

If a control does not respond to the click command, the flag -raw can be passed in. This procedure causes SecureLogin to emulate the mouse and send a direct click message to the control. When you set the #ID to 0, SecureLogin sends the Click command to the window that the script is running on. Also, you can set an X and Y coordinate. The coordinate is relative to the client area of the application, not the screen.

Click #1

Click #0 -raw -x 10 -y 15

Example Windows Script

```
Dialog
Title "Sign On"
EndDialog
PickListAdd User1 Tim
PickListAdd Admin
PicklistDisplay ?User "Please Select your user account"
noedit
SetPlat "?User"
WaitForFocus #105
changepassword "$Username" random
type "$Username"
delay 200
Click #107 -raw -x 10 -y 3
type "$Password"
type "\Alt+s"
```

NOTE: The PicklistDisplay line ends with noedit; noedit is not a separate line.

Ctrl

Usage: CTRL <ctrl-id> [<regex>]

Used in: Windows

Command Type: Dialog specifier

Arguments:

ctrl-ID - The dialog ID of the control to be checked.

<text> - (Optional) The regular expression to match against the control's text.

Remarks:

The Ctrl command checks that the window contains a certain control. This command takes a #ID argument. The #ID is the control ID of a certain control. This number is associated with the control at the time the program is compiled and is constant.

NOTE: This number may change between versions of the third-party software

To determine this number, you can use Window Finder. (See [“Finding Dialog Control IDs” on page 54.](#)) If the control is not found, the script skips to the next Dialog statement.

Example Windows Script

```
Dialog
Title "Enter Network Password"
Ctrl #1218
Ctrl #1041
Ctrl #1
Setplat #1041 "(.*)"
EndDialog
Type "$Username" #1218
Type "$Password" #1219
Click #1
Setprompt "Enter your userid and password for this web site"
```

Delay

Usage: DELAY *period*

Used in: Windows, Terminal Launcher

Command Type: Action

Arguments:

period - A time period, in milliseconds, to pause execution for.

Remarks:

The Delay command delays the action of the script for the time specified. The time is in milliseconds. This delay is in case an application happens to have an introduction screen or some other custom feature. If a script does not work the first time, adding a delay should be your first action. (For example **Delay 5000** will cause a pause for 5 seconds).

Example Windows Script

```
Dialog
Title "Oracle Applications"
Class "AwtFrame"
Delay 2000
EndDialog
delay 2000
Type "$UserName"
delay 800
Type "\N"
Delay 400
Type "$Password"
Type "\N"
```

Dialog / EndDialog

Usage: Dialog

Usage: EndDialog

Used in: Windows

Command Type: dialog specifier

Arguments: None

Remarks:

The Dialog command identifies the beginning of the dialog block. The Dialog/EndDialog statement is used to construct a dialog specifier block that consists of a series of dialog specifier statements (for example, Ctrl, Title).

When a Dialog/EndDialog block is executed, the block executes each of the dialog specifier statements in turn. If the statement succeeds, execution

continues. If a statement fails, the whole dialog/EndDialog block fails and execution skips to the next Dialog/EndDialog block, if any.

The EndDialog command identifies the end of a dialog block. The script that follows the EndDialog command is called the script body. Another Dialog statement or the end of the script terminates the script body.

Example Windows Script

```
Dialog
Title "Enter Network Password"
Ctrl "#1218"
Ctrl "#1"
EndDialog
if ?aaa eq "firstrun"
set ?aaa "blank"
endscript
else
set ?aaa "firstrun"
endif
Type "$Username"#1218
Type "$Password"#1219
Click #1
```

DisplayVariables

Usage: DISPLAYVARIABLES

Used in: Windows, Terminal Launcher

Command Type: Action

Arguments: None

Remarks:

The DisplayVariables command displays a dialog box for the user. This box lists all the user's variables for the current platform. The variables can then be edited to contain their correct values.

For example, if the login is unsuccessful due to an incorrect username or password, the user can be prompted to change these values and continue the login process.

Example Terminal Launcher Script

```
waitfortext login
SetPrompt "Enter Username"
type $Username
type @E
SetPrompt "Enter Password"
type $Password
type @E
delay 2000
repeat 100
iftext "Login incorrect"
messagebox "Your Username or Password Is Invalid"
displayvariables
type $username
type @E
type $password
type @E
endif
```

EndScript

Usage: EndScript

Used in: Windows, Terminal Launcher

Command Type: Action

Arguments: None

Remarks:

Immediately terminates execution of the script.

Example Windows Script

```
restrictvariable ?newPassword "Default Policy"
dialog
class UniCanvas
delay 200
ctrl #8269760
enddialog
endscript
dialog
class UniCanvas
ctrl #8263376
ctrl #8264816
ctrl #8265984
```

```

enddialog
SetPlat "?user"
type $password
type \t
changePassword ?newPassword random
type ?newPassword
type \t
type ?newPassword
delay 1500
type \Ctrl+a
Set $password ?newPassword
dialog
class UniCanvas
delay 200
ctrl #8262512
enddialog
SetPrompt "Enter Username"
type $username #8261552
type $username #8260304
SetPrompt "Enter Password"
delay 500
type $password
type "\n"
SetPrompt "Please enter your Login Details"

```

If / Else /EndIf

Usage: *If operator*

Usage: Else

Usage: EndIf

Used in: Windows

Command Type: Flow control

Arguments:

operator

-gt The left value is greater than the right value.

-eq - The two values are equal.

-lt - The left value is less than the right value.

-text - Provides a shortcut to evaluating text of windows.

Remarks:

The If command establishes a block to be executed if the operator is found to be true.

The Else command works inside an IF block as the script block to execute if the operator evaluates False. The EndIf command is used to terminate the IF block.

The operator eq allows two pieces of data to be compared.

As the following script illustrates, the first argument must be a variable. The second can be any variable or string.

```
If ?result eq "Yes"
#do something
else
#do something else
Endif
```

As the following script illustrates, the operator -text argument works as a shortcut to evaluating text of windows.

```
If -text "Enter Details" #1045
#do something
else
#do something else
Endif
```

Example Script Section

```
if ?aaa eq "firstrun"
set ?aaa "blank"
endscript
else
set ?aaa "firstrun"
endif

# check the version of SecureLogin is greater than v2.3.08
if ?SYSVERSION lt 2308
    messagebox "This version of SecureLogin does not support
    the necessary commands"
    endscript
endif
```

For a list of internal variables that can be used in the SecureLogin script language, see ["Substituting Variables" on page 114](#).

IfText / EndIfText

Usage: IfText *watch-text* EndIfText

Used in: Terminal Launcher

Command Type: Flow control

Arguments:

watch-text - The text to check for.

Remarks:

The IfText command does a case sensitive search of the current terminal window for watch-text. If the text is found, the execution continues to the next line. Otherwise, execution jumps to the statement following the matching EndIfText command. IfText/EndIfText blocks can be nested.

If the text specified in the IfText statement can be found, the IfText command causes Terminal Launcher to execute all of the commands following it until Terminal Launcher reaches the corresponding EndIfText command. Each IfText command must be matched by a corresponding EndIfText command.

These blocks can be placed within each other. It is also useful to place repeat loops around IfText blocks.

The IfText command is mainly used for password changes in terminal applications.

Example Terminal Launcher Script

```
WAITFORTEXT "Enter USERID"
TYPE $Username
TYPE @E
REPEAT 100
IFTEXT "password"
TYPE $Password
TYPE @E
BREAK
ENDIFTEXT
IFTEXT "USERID Expired"
MESSAGEBOX "Your USERID has expired. Please contact the
helpdesk."
ENDSCRIPT
ENDIFTEXT
DELAY 50
ENDREPEAT
```



```
REPEAT 100
IFTEXT "User logged in successfully"
ENDSCRIPT
ENDIFTEXT
ENDREPEAT
```

Increment / Decrement

Usage: Increment *?variable*

Usage: Decrement *?variable*

Used in: Windows

Command Type: Flow control

Arguments:

?variable - The name of the run time variable used to control the count process.

Remarks:

The Increment and Decrement commands are used for flow control to count the number of passes a particular script has made. After the number of instances being checked for has been reached, the script can be instructed to run another task or end the script.

This instruction can be particularly useful in the following situations:

- ◆ When you configure an application whose login panel is similar to other windows within the application.
- ◆ To easily control the number of attempts a user can have to access an application.

Example Windows Script

```
Dialog
class "Button"
title "&OK"
Parent
Title "Password Verification"
Parent
Title "XXXXXX"
class "#32770"
EndParent
type "$Password" #1159 password
```

```
EndParent
EndDialog
type "\N "

repeat 100
if ?count eq 15
type -raw "hello world"
set ?count <NOTSET>
endscript
endif
increment ?count
delay 40
endrepeat
```

KillApp

Usage: KillApp *process-name*

Used in: Windows, Terminal Launcher

Command Type: Flow control

Arguments:

process-name - The name of the process to stop.

Remarks:

You can use the KillApp command to kill an application that did not close successfully since the last time it was run.

Example Windows Script

```
dialog
ctrl #1218
ctrl #1219
ctrl #1041
setplat #1041 "(.*)"
enddialog
if -text "" #1218
type $username #1218
type $password #1219
click #1

else
if ?runCount eq "<NOTSET>"
set ?runCount "0"
endif
```

```

if ?runCount eq "2"
MessageBox "You have entered the wrong details, please contact
administrator"
KillApp "iexplore.exe"
endscript
endif

displayvariables "Please Re-Enter Your Login Details"
type $username #1218
type $password #1219
click #1

if ?runCount eq "0"
set ?runCount "1"
endscript
endif

if ?runCount eq "1"
set ?runCount "2"
endscript
endif

```

MessageBox

Usage: `MessageBox flag ?variable data [data]`

Used in: Windows, Terminal Launcher

Command Type: Action

Arguments:

flag - The -yesno flag allows the user to select either Yes or No instead of clicking the OK button.

?variable - This run time variable can be used to produce a result for the -yesno flag.

data - The text to be displayed to the user.

Remarks:

The MessageBox command pops up a graphical box with text that is passed in. The script is suspended until the user reacts to this message. As the following line illustrates, the MessageBox command can take any number of text arguments, including variables.

```

MessageBox "The User" $Username "has just been logged into the
system"

```

A -yesno flag can also be set when calling a MessageBox. If this flag is set, MessageBox prompts the user with a box that has a Yes and No button rather than OK. As the following line illustrates, passing in a run time variable can capture the result of this box immediately after the flag.

```
MessageBox "Do you wish to continue?" -yesno ?result
```

This option sets the value of ?result to Yes if the user clicked Yes and to No if the user clicked No.

Example Windows Script

```
Dialog
Title "ANR Login Panel"
Ctrl "#1010"
Ctrl "#1030"
EndDialog
Type "$Username" "#1010"
Type "$Password" "#1030"
Type "\N"
Delay 5000
iftext "Login incorrect"
messagebox "Login Incorrect. Updates Login Details Now?" -
yesno ?result
If ?result eq yes
displayvariables
Type "$Username"#1010
Type "$Password"#1030
Type "\N"
Else
KillApp "anr32.exe"
Break
```

Parent / EndParent

Usage: Parent

Usage: EndParent

Used in: Windows

Command Type: Flow control

Arguments: None

Remarks:

The Parent command begins a Parent Statement that contains information about the window's parent. The commands that follow work the same in a parent block as they do in a dialog block. If they evaluate to False, the script ends. However the commands query the parent window.

For example, the Title command in a parent block returns False if the title of the parent does not match the one specified in the command.

Parent blocks, however, do not skip to the next parent block like a dialog block does. Instead, it immediately returns the error to the dialog block, and the dialog block then continues to the next dialog block or ends if no more dialog blocks exist.

As the following lines illustrate, the Parent command can also be nested:

```
Dialog
Parent
Parent
Title "Login"
EndParent
EndParent
```

This nesting allows SecureLogin to check the parent of the parent.

As the following script illustrates, you can also use the Parent command to set the subject of the script to the parent window, allowing use of the other commands such as Type and Click.

```
Dialog
Title "Login"
EndDialog
Type "$username"
Parent
Click #24
EndParent
Click #4
```

If the Parent command is used on a window that has no parent, the Script block breaks. The block then skips to the next dialog block or ends if no more blocks exist.

The EndParent command terminates a Parent block and sets the subject of the script back to the original window.

PickListAdd

Usage: PickListAdd *display-text return-value*

Used in: Windows

Command Type: Action

Arguments:

display-text - The text that will be displayed in the picklist for the specified option.

return-value - The value returned from the picklist.

Remarks:

The PickListAdd command allows users who have multiple accounts for a particular system to choose which accounts they use to log in with. It can also be used for users who have one Mainframe account, with multiple sessions to choose from.

The PickListAdd command adds a variable to this list for the user to pick from. The first string passed in is the *display-text*. This is the text as it is displayed in the picklist. The second string is the *return-value* that is supplied from the PickListDisplay command.

If the second string is not passed in, the *display-text* is returned.

The following script will display AdminUser and User1 when PickListDisplay is called.

```
PickListAdd AdminUser cn=administrator  
PicklistAdd User1 cn=tim
```

For more information on the PickListDisplay command, see [“PickListDisplay” on page 95](#).

Example Terminal Launcher Script

```
PICKLISTADD "Open ISPF" "ispf"  
PICKLISTADD "TSO"  
PICKLISTDISPLAY ?app "Please select which application you  
would like to log in to." noedit  
waitfortext USERID  
type $Username  
type @E  
waitfortext Password
```

```
type $Password
type @E
waitfortext "Enter command"
type ?app
type @E
```

PickListDisplay

Usage: PickListDisplay *?variable display-text flag*

Used in: Windows

Command Type: Action

Arguments:

?variable - The output variable for the selected option.

display-text - The description text for the picklist box.

flag - The noedit flag prevents users from adding variables.

Remarks:

The PickListDisplay command displays the list of options built by previous calls to PickListAdd. PickListDisplay then returns the result in a variable passed into the command. If none of the displayed entries are the desired account, users can enter their own data in an edit field at the bottom of the picklist.

As the following line illustrates, you can turn off this feature by setting the noedit flag:

```
PickListDisplay ?result "Please Pick your user" noedit
```

Example Script Section

```
PICKLISTADD "option 1" value1
PICKLISTADD "option 2" value2
PICKLISTDISPLAY ?sel1 "Please choose an option from this first
picklist box"
PICKLISTADD "option 1b" value1b
PICKLISTADD "option 2b" value2b
PICKLISTDISPLAY ?sel2 "Please choose an option from this
second picklist box" noedit
```

ReadText

Usage: (Windows Type) ReadText #*ID variable*

Usage: (Terminal Launcher Type) ReadText *variable column-number start-position character-number*

Used in: Windows, Terminal Launcher

Command Type: Action

Arguments:

#ID - The control ID of the text to be read.

variable - The variable that will receive the text that is read.

column-number - The column number to begin the read from.

start-position - The position number of the first character to be read.

character-number - The number of characters to be read.

Remarks:

The ReadText command runs in both Windows and Terminal Launcher type scripts. Although the arguments for each type are different, the result of each command is the same.

In a Windows script, the ReadText command reads the text from any given Control ID and passes it to the specified variable. For this command to function correctly, the Control ID of the text being read must be valid.

In the Terminal Launcher script, the ReadText command reads in a specified number of characters, starting at a given point, and passes the characters to the variable.

Example Terminal Launcher Script

```
WAITFORTEXT "USERID
TYPE @0
TYPE $userid
TYPE @T
TYPE $password
TYPE @E
Delay 250
If -Text "TSO/E LOGON"
Delay 250
```



```
ReadText ?TSOMsg 3 2 1
If ?TSOMsg eq "IKJ"
Else
TYPE @0
TYPE $password
TYPE @E
EndIf
```

RegSplit

Usage: RegSplit *regex input-string output-string1 output-string2*

Used in: Windows

Command Type: Action

Arguments:

regex - the regular expression

input-string -

output-string1

output-string2

Remarks:

The RegSplit command allows a string to be split by using a regular expression. This command is similar to Setplat's regular expression feature. Output string1 contains the first sub-expression, OutPut string2 contains the second sub-expressions, and so on

The following illustrates the RegSplit command:.

Regsplit "The user (.*) has logged into the system at (.*)" "The user Tim has logged into the system at 204.223.245.2" ?name ?system.

Now the variable *?name* will contain "Tim" and the variable *system* will contain 204.223.245.2.

Example Windows Script

```
dialog
title "Username and Password Required"
ctrl #331
ctrl #214
ctrl #330
```

```

enddialog
readtext #331 ?text
regsplit "Enter username for (.*) at (.*):" ?text ?realm ?site
index ".bt.com"?site ?position
if ?position eq "0"
endscript
endif
strcat ?platform ?site " - " ?realm
setplat ?platform
type #214 $username
type #330 password
click #1

```

NOTE: The line beginning with “regsplit” ends with “?site.”

Repeat

Usage: Repeat *loops*

Used in: Windows, Terminal Launcher

Command Type: Action

Arguments:

loops - The number of repeat associated with a script block.

Remarks:

The Repeat command establishes a script block similar to the If command. The repeat block is terminated by an EndRepeat command and can be broken out of by use of the Break command. The loops argument is the number of times the block is to be repeated before continuing on in the script.

Example Terminal Launcher Script

```

waitfortext login
SetPrompt "Enter Username"
type $Username
type @E
repeat 1000
iftext "Password"
delay 200
SetPrompt "Enter Password"
type $Password
type @E
delay 2000
repeat 100

```

```
iftext "Enter login password"
type $Password
type @E
WaitForText "New password"
changepassword $Password
type $Password
type @E
waitfortext "Re-enter new password"
type $Password
type @E
waitfortext "passwd successfully changed"
set ?tempPassword $Password
break
endiftext
```

RestrictVariable

Usage: RestrictVariable *variable-name password-policy*

Used in: Windows, terminal Launcher

Command Type: Action

Arguments:

variable-name - The name of the variable to restrict.

password-policy - The name of the policy to restrict the variable.

Remarks:

The RestrictVariable command monitors a variable and enforces a specified password policy on that variable.

For more information on the RestrictVariable command, see [Chapter 6, "Using Password Policies,"](#) on page 71.

Example Terminal Launcher Script

```
restrictvariable $Password passpolscript
waitfortext USERID
type $Username
type @E
waitfortext password
type $Password
type @E
repeat 1000
iftext "password expired"
```

```

change password $Password random
type ?Password
type @E
waitfortext "confirm new password"
type ?Password
type @E
break
endiftext
iftext "User logged in successfully"
break
endiftext |
delay 10
endrepeat

```

Run

Usage: Run *command* [*arg1* [*arg2*] [...]]

Used in: Windows, Terminal Launcher

Command Type: Action

Arguments:

command - The full path to the command to be executed.

arg1, *arg2* - A list of arguments to the command.

Remarks:

The Run command starts the program specified in <command> and then uses the specified arguments. The script does not wait for the launched program to complete.

Example SecureLogin Startup Script

```

MessageBox "Good Morning" "%CN" "Would You Like To Start Notes
Now?" -yesno ?Result
If ?Result eq yes
Run "c:\lotus\notes\notes.exe"
Else
EndScript

```

Strcat

Usage: Strcat *variable input-string1 input-string2*

Used in: Windows

Command Type: Action

Arguments:

variable - The regular expression.

input-string1 - The first data string or variable.

input-string2 - The second data string or variable.

Remarks:

The Strcat command copies one piece of data onto the end of another piece of data.

For example, you include the following line in a script:

```
STRCAT ?result SecureRemote $username
```

When \$username is Tim, the ?result contains "SecureRemote Tim".

Example Script

```
dialog
title "Username and Password Required"
ctrl #331
ctrl #214
ctrl #330
enddialog
readtext #331 ?text
regsplit "Enter username for (.*?) at (.*):" ?text ?realm
?site
index ".bt.com" ?site ?position
if ?position eq "0"
endscript
endif
strcat ?platform ?site " - " ?realm
setplat ?platform
type #214 $username
type #330 password
click #1
```

NOTE: The line beginning with "regsplit" ends with "?site".

Set

Usage: Set *variable data*

Used in: Windows

Command Type: Action

Arguments:

variable - The variable being assigned to.

data - The text or variable being read from.

Remarks:

The Set command copies the value of *data* into *variable*. The *data* argument can be any text or another variable. However, the *variable* argument must be either ?variable or \$variable.

Example Windows Script

```
RestrictVariable ?newPassword "Default"
Dialog
Title "Lotus Notes"
set ?PasswordExpired "FALSE"
Ctrl #65535 "WARNING: Your password will expire on"
EndDialog
set ?PasswordExpired "TRUE"
MessageBox "hello"
Dialog
class "#32770"
ctrl #17 "Set &Password..."
EndDialog
if ?PasswordExpired eq "TRUE"
changepassword ?newPassword random
set $newPassword ?newPassword
click #17
set ?PasswordExpired "FALSE"
endif
MessageBox "Hello"
Dialog
Title "Set Password"
EndDialog
type ?newPassword #1005
click #1
```

SetCursor

Usage: SetCursor *screen-offset*

Used in: Terminal Launcher

Command Type: Action

Arguments:

screen-offset - The position to which the cursor should be moved.

Remarks:

The SetCursor command sets the cursor to a particular position. The position number must be a number greater than 0 (for example SETCURSOR 200). If the screen position is invalid, Terminal Launcher displays an error message.

Example Terminal Launcher Script

```
waitfortext "Userid"
setprompt "Please enter your mainframe login name"
type $USERNAME
type @T
setprompt "Please enter your mainframe password" password
type $password
type @E

waitfortext "MAIN MENU"
setcursor 2415
type "start Session A"
type @E
```

SetFocus

Usage: SetFocus *#ID*

Used in: Windows

Command Type: Action

Arguments: None

Remarks:

The SetFocus command gives the keyboard focus to a specified Control ID.

For the SetFocus command to function correctly, the Control IDs that are specified must be valid.

Example Windows Script

```
Dialog
Title "Acct Statement Login"
Ctrl "#160"
Ctrl "#170"
Class "DialogClass32"
EndDialog
Delay 500
SetFocus "#160"
SetPrompt "Enter UserName"
Type "$Username" "#160"
Type "\N"
SetPrompt "Enter Password"
Type "$Password" "#170"
Click #4
SetPrompt "Please Enter Your Login Details For The Acme
Accounting System"
```

SetPlat

Usage: SetPlat *platform-name*

Usage: SetPlat *regex #ID*

Used in: Windows, Terminal Launcher

Command Type: Action

Arguments:

platform-name - The platform name to read the variables from.

regex - A regular expression to be used as the platform name.

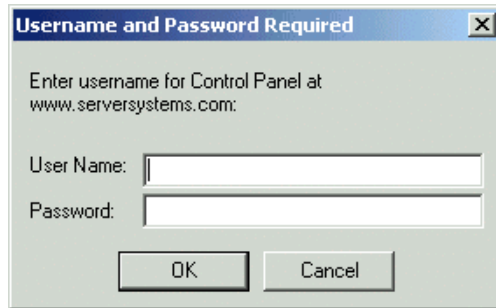
#ID - The control ID of the regular expression to be used.

Remarks:

The SetPlat command changes the platform that variables are read and saved to. If an application has a password in common with another application, instead of the user entering it twice you can SetPlat "Second platform". SecureLogin then switches the script engine to read variables to the other platform. If the platform does not exist, it will be created.

SetPlat can also take a #ID to read from and support regular expressions.

For example, the following figure illustrates a standard dialog box for accessing a password-protected site using Netscape* Navigator.



The following script illustrates a simple way to sign in to this site, using a platform called netscape.exe:

```
Window "Username and Password Required"  
  Type $username #214  
  Type $password #330  
  Click #1
```

However, this script limits users to one username/password combination for all http-authentication Web sites accessed through Netscape. Such a limitation poses a significant problem.

Notice that the dialog box pictured above always uses the same format of text, which contains the name of the Web site to sign in to. The solution to this problem is to use a dialog block with a SetPlat statement similar to the following:

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

The power of this script is in the following line:

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

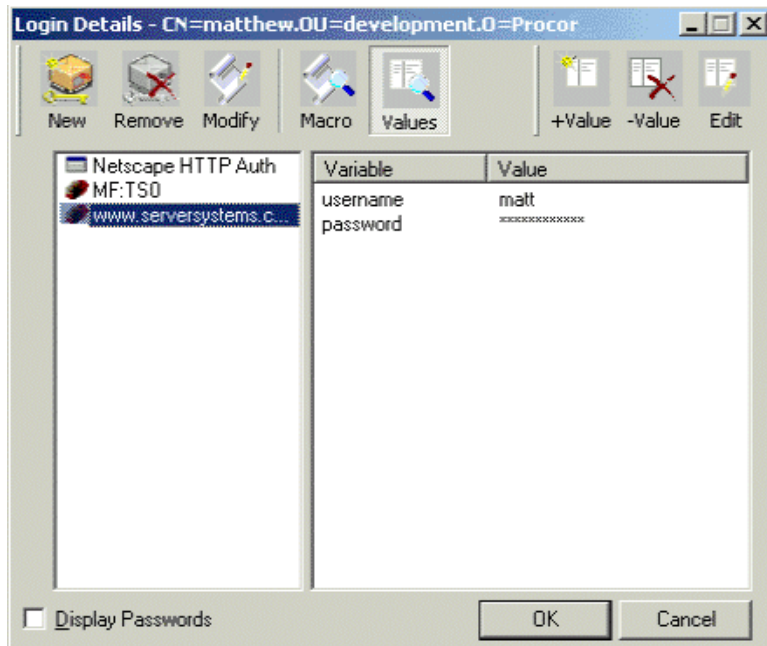
The script first reads the following line from dialog control ID 331:

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

The script then applies the regular expression to this text. Regular expressions are a powerful way to manipulate text strings. However, for most purposes a few very basic commands will suffice:

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

The text matched inside the parentheses then becomes the symbol platform. In the preceding example script, after the user has run the script you will see the username and password saved under `www.serversystems.com`.



If a dialog control ID is not specified to Setplat, the symbol platform will be unconditionally be changed to the platform specified in <regular-expression>. An unconditional Setplat command is only valid if specified before dialog/ enddialog statements.

For more information on the SetPlat command, refer to [“PickListAdd” on page 94](#).

Setprompt

Usage: Setprompt *prompt-text*

Used in: Windows, Terminal Launcher, Web

Command Type: Action

Arguments:

prompt-text The information displayed at the top of the prompt variable box.

Remarks:

The SetPrompt command customizes the text in the variable boxes used when prompting for new variables or when DisplayVariables command is used.

To replace the text at the top of the box, place the SetPrompt command at the very bottom of the script.

To replace the text next to any variable in a box, place the SetPrompt command immediately before the variable in the script.

Example Windows Script

```
Dialog
Title "Login"
Ctrl "#1060"
Ctrl "#1061"
EndDialog
SetPrompt "Enter UserName"
Type "$Username"#1060
SetPrompt "Enter Password"
Type "$Password"#1059
Click #1061
SetPrompt "Please Enter Your SecureRemote Login Details"
```

Title

Usage: Title *window-title*

Used in: Windows

Command Type: Dialog specifier

Arguments:

window-title - The text to test against the window title.

Remarks:

The Title command retrieves the title of the window and compares it against the string passed in as the argument. For this section of the script to run, the two must match.

The following line illustrates the Title command:

```
Title "Secure Remote Login"
```

You can use Window Finder to locate the window title. See [“Finding Dialog Control IDs” on page 54](#).

Example Windows Script

```
Dialog
Title "SecureRemote Login"
Ctrl "#1060"
Ctrl "#1061"
EndDialog
Type "$Username"#1060
Type "$Password"#1059
Click #1061
```

Type

Usage: Type *flag text [ctrl-ID]*

Used in: Windows, Terminal Launcher, Web

Command Type: Action

Arguments:

flag - “-raw” (Only available when not using Control IDs)

text - The text to type into this area.

ctrl-ID - The dialog ID of the control to be checked.

Remarks:

The Type command is the command that tells SecureLogin to type something into a window. The first argument is text. It is simply a string enclosed in quotation marks (" "). The ctrl-ID argument is the identifier of the control that the text is to be typed into.

NOTE: If the ctrl-id is not specified, SecureLogin will send keystrokes to the control that currently has focus. This can be useful if a special control needs to be typed into. SecureLogin provides custom keys for Tab and Enter. The script parser interprets these escape sequences at run time.

This command is used to enter data, such as username and passwords, into windows. It can take a #ID as an argument after the data to enter. If the #ID is passed in, the information is sent directly to the window. If no #ID is passed in, SecureLogin emulates the keyboard directly and sends individual keystrokes.

The flag for the Type command is -raw. It is only supported when you don't use the Control ID. By default, when you enter data into a window, SecureLogin verifies that the window exists, before continuing the script. This checking is disabled when the -raw flag is set, causing SecureLogin to log in to whatever window has focus. This can be useful if you need to use a delay after a step in the login process, and need to enter data after this delay.

For example, the Type command can be used to submit Command keys to an application that requires the Alt and L keys to be pressed.

For more information on Type commands, see [“Windows and Internet Keystroke Commands” on page 121](#).

Example Windows Script

```
Dialog
Title "Lotus Notes"
CTRL #65535 "WARNING: Your password will expire on (.*)"
EndDialog
set ?PassExpired "YES"
click #2
Dialog
  title "HPInboxHeader"
  class "NotesSubprog"
EndDialog
```

```

if ?PassExpired eq "YES"
  # Select File, Tools, User ID, Set Password.
  Type "\Alt+F"
  Type "T"
  Type "I"
  Type "P"
EndIf
Dialog
title "Set Password"
ctrl #311 "Enter your new password:"
EndDialog
if ?RunSETPWDBefore eq "YES"
  Type ?Napes #301
  Set ?RunSETPWDBefore "NO"
  Click #1
  Set $Password ?NewPass
  EndScript
endif
ChangePassword ?NewPass random
Type ?NewPass #301
Click #1
Set ?RunSETPWDBefore "YES"
Dialog
Title "Lotus Notes"
ctrl #65535 ""You have used this password before. Please
choose a new one"
EndDialog
Set ?RunSETPWDBefore "NO"

```

WaitForFocus

Usage: WaitForFocus ctrl-ID *repeat-loops*

Used in: Windows

Command Type: Action

Arguments:

#ID - The control ID of the box to wait for.

repeat-loops - The number of repeat loops that will run.

Remarks:

The WaitForFocus command suspends the running of the script until the control has received keyboard focus or until the repeat loops expire. The repeat-loops argument is an optional value that defines how many loop cycles

to wait. The value defaults to 3000 loops if nothing is set. As soon as focus is received, the script continues.

As the following line illustrates, you can set the repeat loop to never expire by setting the value to a negative number:

```
WaitForFocus "#1065" "-1"
```

If the Control ID is set to 0, it will loop until the window defined in the Dialog / EndDialog statement is given keyboard focus.

NOTE: Do not place WaitForFocus commands within Dialog / EndDialog statements.

Example Windows Script

```
Dialog
Title "Login"
Class "#32770"
Enddialog
WaitForFocus #600
delay 200
SetPrompt "Enter Username"
Type $Username
Type \t
SetPrompt "Enter Password"
Type $Password
delay 500
type \n
Dialog
Title "Login Error"
EndDialog
MessageBox "The username or password you have entered are
incorrect"
displayvariables
Click #1
SetPrompt "Please Enter your context Login details"
```

Waitfortext

Usage: Waitfortext *text*

Used in: Terminal Launcher

Command Type: Flow control

Arguments:

text - The text to wait for.

Remarks:

The Waitfortext command causes Terminal Launcher to wait for the specified text to be displayed before continuing. This command is essential because the user will always want to wait for a particular text to be displayed on the screen before continuing. Otherwise, text may be typed too soon.

The text that users wait for may appear anywhere on the terminal screen. The text is case sensitive. If the text is written in the wrong case, Terminal Launcher waits on that screen until it times out, unable to find the correct text.

Example Terminal Launcher Script

```
restrictvariable ?newPassword "Default Policy"
waitfortext login
SetPrompt "Enter Username"
type $Username
type @E
Set ?tempPassword $Password
repeat 1000
iftext "Password"
delay 200
SetPrompt "Enter Password"
type $Password
type @E
delay 2000
repeat 100
iftext "Enter login password"
type $Password
type @E
WaitForText "New password"
changepassword $Password
type $Password
type @E
waitfortext "Re-enter new password"
type $Password
type @E
waitfortext "passwd successfully changed"
set ?tempPassword $Password
break
endiftext
```


A

SecureLogin Script Language

Structuring and Executing Scripts

A script is a simple piece of text that is stored by the SecureLogin script broker. Each script has a name, called the platform name, which uniquely identifies it within a particular single sign-on database.

In addition, each script has a type, known as the platform type. The platform type specifies the type of application the script refers to and hence which of the SecureLogin components executes it.

SecureLogin scripts execute sequentially from the first line. There are no flow control mechanisms as such. There are, however, instances where a component may choose not to execute certain statements, as in the Dialog/EndDialog statement. Each line in the script consists of one or more arguments.

Arguments are separated by white space (spaces and tabs), unless they are enclosed with quotation marks. For example, the following line contains three arguments:

```
A simple "command to get started"
```

- ◆ A
- ◆ simple
- ◆ "command to get started"

After a script has been broken into arguments, the quotation marks are removed. If you need to specify an actual quotation mark in a script, precede it with a `\` (for example `\"`).

The first argument on a line is the command. It specifies the action the line takes. (See [Chapter 7, "SecureLogin Commands," on page 77](#).) The rest of the arguments on the line, if any, are passed to that command. Different commands take varying numbers of arguments.

If a line begins with a # character, the line is treated as a comment and is ignored in the script language. The following example illustrates this character.

```
Window "login"  
Delay 30  
# this delay is so the window is  
# created correctly  
Type "$Username"
```

Scripts are interpreted as SecureLogin components perform the sign-in process. This functionality ensures that any variables that are substituted or passtickets that are generated are current.

Using Variables

Substituting Variables

For the purposes of the script language, a variable is any identifier preceded by one of the characters %, \$ or !. This identifier is called the variable prefix (for example, %CN, \$username and !default). At run time, these identifiers are substituted for their values, which the script parser works out on each run.

The variable prefix determines the type of the value and the steps that should be taken to determine it. The three variable types are Directory Attribute (%), Symbol Table (\$), and Vasco Pass Ticket (!).

SecureLogin supports a list of internal variables that the script language may used to

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

Variable Name	Description
?SYSVERSION	The local SecureLogin windows agent version. This variable can be used to determine if specific support is built into the product running on the user's workstation. The format of the variable is major.minor.build (for example, 2030008, which represents v2.30.008)
?SYSUSER(<i>system</i>)	This variable is the name of the user that was last used in the GINA or Windows* 95/98 login panel. This variable will only be available when the SecureLogin login extension is installed.

Variable Name	Description
?SYSPASSWORD(<i>system</i>)	This variable is the password that matches the username presented in the GINA panel. This value can only be retrieved if the SecureLogin login extension is installed in the GINA or Windows 95/98 login panel.
?SYSCONTEXT(<i>system</i>)	This variable lists the NDS eDirectory™ user context as entered in the GINA or Windows 95/98 login panel. This variable requires the login extension to be installed.
?SYSTREE(<i>system</i>)	This variable is the NDS eDirectory tree name that was entered by the user. This variable requires the login extension to be installed.
?SYSSERVER(<i>system</i>)	This variable is the name of the server that was entered in the Login GINA or Windows 95/98 login panel. This variable requires the login extension to be installed.
?BROWSERTYPE(<i>system</i>)	This variable contains either Internet Explorer or Netscape* and indicates which browser the script is running.

Using eDirectory Attributes

This type of variable is prefixed by the % character. It instructs the parser to read a value from the current User's object in NDS eDirectory.

The attribute read must be a string, either case sensitive or not case sensitive. The attribute is read each time that the script is run.

Using Variables

A variable prefixed with the \$ character is looked up in the current application entry.

If the value that was substituted also has a \$ as its first character, that value is also run through the substitution engine. However, this process only occurs once.

If the symbol is not found in the current application, the user is prompted with a dialog box to enter the variable's value.

If the user enters a value, it is substituted immediately and the value is saved.

Using the Passticket Attribute

The ! character specifies that a one-time password should be generated.

To generate a passticket, a DES key and offset are required. At run time, the parser looks for the variables DESKEY and DESOFFSET in the platform specified as the rest of the variable identifier.

For example, the variable !Passticket looks for the variables DESKEY and DESOFFSET in the Passticket platform. The special identifier !default reads these variables from the current platform.

If either DESKEY or DESOFFSET is not found, random values are generated and saved in the designated platform.

Each platform script has an associated symbol table. A symbol table is referenced by its platform name and contains a list of variables and values that can be used in the script.

B

Script and Keystroke Commands

Terminal Launcher HLLAPI Commands

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

The table in “[Script Reference Guide](#)” on page 118 lists the @ commands that you can use in the SecureLogin script Type command. These commands perform specific emulator and mainframe functions. For example, you can send an Enter, tab key, or cursor key.

You can also send functions. For example, you can issue a mainframe emulator print screen or reset function.

The @ commands are used in script language. They have special meaning in the Type *@command* format.

The following example illustrates @ commands:

Script	Description
WAITFORTEXT “Login:” Type \$username	
Type @T	Sends the Tab key after the \$username variable.
Type \$password	
Type @E	Sends an Enter key after the \$password variable.

Script Reference Guide

The Type Command	Meaning	The Type Command	Meaning
@B	Left Tab	@A@C	Test
@C	Clear	@A@D	Word Delete
@D	Delete	@A@E	Field Exit
@E	Enter	@A@F	Erase Input
@F	Erase EOF	@A@H	System Request
@H	Help	@A@I	Insert Toggle
@I	Insert	@A@J	Cursor Select
@J	Jump (Set Focus)	@A@L	Cursor Left Fast
@L	Cursor Left	@A@Q	Attention
@N	New Line	@A@R	Device Cancel (Cancels Print Presentation Space)
@O	Space	@A@T	Print Presentation Space
@P	Print	@A@U	Cursor Up Fast
@R	Reset	@A@V	Cursor Down Fast
@T	Right Tab	@A@Z	Cursor Right Fast
@U	Cursor Up	@A@9	Reverse Video
@V	Cursor Down	@A@b	Underscore
@X*	DBCS (Reserved)	@A@c	Reset Reverse Video
@Y	Caps Lock (No action)	@A@d	Red
@Z	Cursor Right	@A@e	Pink
@0	Home	@A@f	Green
@1	PF1/F1	@A@g	Yellow
@2	PF2/F2	@A@h	Blue

The Type Command	Meaning	The Type Command	Meaning
@3	PF3/F3	@A@i	Turquoise
@4	PF4/F4	@A@l	Reset Host Colors
@5	PF5/F5	@A@j	White
@6	PF6/F6	@A@t	Print (Personal Computer)
@7	PF7/F7	@A@y	Forward Word Tab
@8	PF8/F8	@A@z	Backward Word Tab
@9	PF9/F9	@A@-	Field -
@a	PF10/F10	@A@<	Record Backspace
@b	PF11/F11	@A@+	Field +
@c	PF12/F12	@S@x	Dup
@d	PF13	@S@E	Print Presentation Space or Host
@e	PF14	@S@y	Field Mark
@f	PF15	@X@c	Split Vertical Bar (!)
@g	PF16	@X@7	Forward Character
@h	PF17	@X@6	Display Attribute
@i	PF18	@X@5	Generate SO/SI
@j	PF19	@X@1	Display SO/SI
@k	PF20	@M@0	VT Numeric Pad 0
@l	PF21	@M@1	VT Numeric Pad 1
@m	PF22	@M@2	VT Numeric Pad 2
@n	PF23	@m@3s	VT Numeric Pad 3
@o	PF24	@M@4	VT Numeric Pad 4
@q	End	@M@5	VT Numeric Pad 5

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

Windows and Internet Keystroke Commands

SecureLogin can send special keyboard keystrokes to Windows* and Internet applications. These keystrokes emulate the user's keyboard entry.

These special commands include the ability to select menu items, send ALT, and send other keyboard combinations.

Keystrokes to Enter	Description
Type "\Alt+x"	The variable x is any key.
Type "\Ctrl+x"	
Type "\Shift+x"	
Type "\T"	Presses the Tab key.
Type "\N"	Presses the Enter key.
Type "\-T"	Presses Shift +Tab keys.
Type "\b"	Presses the backspace key.
Type "\e"	Presses the End key.
Type "\d"	Presses the Delete key.
Type "\h"	Presses the Home key.
Type \$password #1056	Sends the password to the control #1056.
Type "\$Password" password	Ensures that the password is not displayed in clear text when the user enters it.

NOTE: Always add the word "password" to the end of any password variable lines in the script.

To select a menu item within an application, you can use the following sequence:

```
Type \Alt+F (select file)
Type T (select tools)
Type C (select Change Password)
```

For additional information, refer to the Type Command columns in ["Script Reference Guide" on page 118.](#)

